DEPARTAMENTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

INGENIERÍA SUPERIOR EN INFORMÁTICA

PROYECTO DE SISTEMAS INFORMÁTICOS

CURSO 2010/2011

TREEDOC, UN GESTOR DE CONTENIDOS ORIENTADO A LA EDICIÓN COLABORATIVA DE DOCUMENTOS

Autores: Gerardo Óscar Jiménez Tornos

Natalia López Romero

Luis Moyano Rufo

Tutor: Rubén Fuentes Fernández

Universidad Complutense Madrid

Facultad de Informática

Los abajo firmantes, matriculados en la asignatura Sistemas Informáticos la Facultad de Informática, autorizan a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes de los autores, la documentación y/o el prototipo desarrollado durante el curso académico 2010-2011 bajo la dirección del Dr. Rubén Fuentes Fernández en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Gerardo Óscar Jiménez Tornos

Natalia López Romero

Luis Moyano Rufo

Contenido

Tabla de ilustraciones

1 Resumen

En el presente documento se describe un sistema de gestión y edición de documentos que facilita el trabajo colaborativo. Se ha optado por una aplicación Web para facilitar el acceso a los usuarios. De esta forma, se permite que varios usuarios editen el mismo documento en el mismo instante, y los únicos requisitos necesarios para ello son un navegador Web y conexión a internet.

Consta con un módulo para la gestión de documentos desde el cual se pueden realizar las acciones básicas como crear, abrir o borrar un documento, y que permite acceder al editor, y otro para la gestión de contactos, que permite acceder a información útil sobre otros usuarios con los que se puede trabajar de forma colaborativa.

Los documentos creados con TreeDoc pueden contener texto, imágenes, mapas, fórmulas, vídeos y enlaces a otros documentos, y una vez que se ha creado el documento, se puede exportar a distintos formatos.

Palabras clave: Sistemas colaborativos, editor, gestión de documentos, colaboración en tiempo real.

2 Abstract

This document describes a collaborative system for document management and edition. Multiple users can collaborate in the edition of the same document at the same time, and the only requirements to access to the application are a Web browser and Internet connection as it has been implemented as a Web application.

The system consists of two modules; one for the documents management, which provides access to the basic functionalities (create, open and delete a document) and to the document editor, and other for the contacts management, which provides useful information about other users that can collaborate.

It is possible to create a document with text, pictures, maps, formulas, videos and links to other documents. It is also possible to export the documents to different formats.

Keywords: Collaborative systems, editor, document management, real-time collaboration.

3 Introducción

TreeDoc es un sistema de gestión de contenidos (*CMS, Content Management System*) orientado a la edición colaborativa de documentos en la web en tiempo real. Este tipo de sistemas permiten que varios usuarios editen un mismo documento desde distintos equipos en el mismo momento, y llevar a cabo la gestión de documentos. Los CMS pueden gestionar distintos tipos de contenidos (desde datos científicos hasta imágenes, vídeos o documentos). Sobre los contenidos el CMS suele proporcionar funcionalidades de almacenamiento y control de los mismos. En cuanto a control se refiere, éste puede ser de distintos tipos entre los que se incluyen el control de versiones y de acceso por los usuarios.

Dentro de este tipo de sistemas, TreeDoc tiene dos características clave. En primer lugar, TeeDoc concibe el documento como un árbol de nodos, cada uno de los cuales es una unidad de información y contenido. En segundo lugar, TreeDoc se ha diseñado con un particular hincapié en separar en la gestión de documentos los contenidos de la presentación. Ésta es una aproximación ampliamente adoptada en populares herramientas como Drupal (1) y que promueven estándares como XML (*eXtensible Markup Language*) (2) y CSS (*Cascading Style Sheets*) (3).

Sobre la base anterior, TreeDoc proporciona facilidades de gestión y edición colaborativa de documentos. Precisamente la edición colaborativa de documentos se ve facilitada por las características mencionadas. Los autores pueden trabajar simultáneamente sobre nodos diferentes, al tiempo que la estructura de árbol facilita comprender las dependencias estructurales de los contenidos. Además, esta estructura simplifica aspectos de presentación como el sangrado y la correcta colocación de los párrafos. Toda esta funcionalidad es accesible a través de una interfaz web que sólo requiere para su uso un navegador.

El resto del documento discute con mayor detalle estos aspectos. Se organiza como sigue. En la sección 0 se realiza una revisión del estado del arte, considerando otros sistemas que pueden tener relación con este proyecto, sus principales características, y los aspectos positivos y negativos de los mismos. La sección 5 presenta la especificación del sistema mediante la descripción de los requisitos identificados a partir de la revisión de la sección 0. La sección 4 describe el diseño del sistema desarrollado de acuerdo con los requisitos de la sección 0. A continuación, la sección 7 describe el uso del sistema para la creación de un documento, la gestión de su acceso y la visualización. Finalmente, la sección 0 discute las conclusiones acerca del trabajo realizado y la sección 0 introduce algunas potenciales líneas de trabajo futuro.

El documento también contiene varios apéndices con información adicional sobre las secciones anteriores. Esta información se centra en los aspectos más técnicos del presente trabajo.

4 Estado del arte

El presente trabajo con CMSs para edición colaborativa está relacionado con varias tendencias y tipos de herramientas de creciente importancia en internet. En concreto, esta sección considera los sistemas para trabajo colaborativo y de gestión de contenidos, y cómo se han adaptado estos a los últimos modelos y tecnologías de la industria, tales como el *cloud computing,* la *Web 2.0* y las aplicaciones RIA (*Rich Internet Application*). La sección comienza con una explicación de estos conceptos, después analiza algunos sistemas concretos que existen en la actualidad y finaliza con las conclusiones extraídas de este análisis inicial acerca de características clave de estos sistemas y tendencias en su desarrollo.

En primer lugar, definiremos qué es un sistema de software colaborativo, o *groupware* y sus características. Según Chaffey se trata de: "Sistemas basados en computadoras que apoyan a grupos de personas que trabajan en una tarea común y que proveen una interfaz para un ambiente compartido" (4). Para alcanzar este objetivo, estos sistemas persiguen tres requisitos clave (5):

- Proporcionar un ambiente de colaboración, en el que realmente se perciba que el trabajo en grupo se lleva a cabo.
- Mantener la información en un solo sitio común para todos los miembros.
- Interactuar con otros usuarios, de forma escrita, mediante voz o mediante videoconferencia.

En función del momento en el que se produce la interacción entre los distintos usuarios que participan en el sistema se puede hablar de dos categorías:

- Sistemas colaborativos asíncronos (*no real-time groupware*): En ellos, los usuarios trabajan de forma colaborativa, pero no en el mismo momento. Concretamente, en los editores de texto colaborativos asíncronos el trabajo suele realizarse de forma secuencial, enviando diferentes versiones a los usuarios. Normalmente los cambios suelen incluir comentarios.
- Sistemas colaborativos síncronos (*real-time groupware*): Los usuarios llevan a cabo la tarea de forma simultánea. Para que los usuarios puedan trabajar en un sistema de este tipo, el tiempo de respuesta debe ser mínimo, proporcionando sensación de que se está trabajando en tiempo real. Además, los usuarios deben percibir las acciones que llevan a cabo otros usuarios. Esta característica se denomina en a*wareness*.

Los CMS pueden ser clasificados como sistemas de software colaborativos porque proporcionan funcionalidades para crear y gestionar los contenidos colaborativamente a los usuarios.

En este proyecto utilizamos además una de las tecnologías que más interés están despertando actualmente: *cloud computing*. Para tener una referencia, este nuevo modelo ha aparecido como muy prioritario (entre los 5 primeros puestos) durante varios años en los informes publicados por *Gartner* sobre tecnologías que tendrán un alto impacto en el futuro próximo (6).

La definición de Cloud Computing publicada por el NIST (*National Institute of Standards and Technology* de Estados Unidos) en 2011 dice sobre el *cloud Computing* que "permite el acceso a una fuente común de recursos a través de la red (por ejemplo redes, servidores, aplicaciones o servicios)" y que estos recursos "se suministran rápidamente, con un esfuerzo de gestión mínimo por parte del proveedor del servicios"(7). Se puede decir que la adopción del *cloud computing* es una tendencia natural de los entornos colaborativos. Permite que los archivos, documentos y cualquier contenido este en "la nube", accesible desde cualquier ordenador que disponga de conexión, lo que permite su compartición y centralizar recursos.

En nuestro caso, se accederá a la aplicación a través de "la nube". De esta forma, será posible que varios usuarios accedan en el mismo momento desde distintos lugares.

La tercera línea relacionada con nuestro trabajo es la denominada "Web 2.0". El término se asocia con las aplicaciones enfocadas a la participación y colaboración en la Web. Concretamente, se conecta con la creación colaborativa de los contenidos, donde la barrera entre autor y consumidor queda difuminada. Las herramientas de este tipo que podemos encontrar son muy variadas: los *blogs*, presentaciones en línea, redes sociales, *wikis,* plataformas educativas o videojuegos son algunas de las más comunes. Entre todas ellas, las *wikis* con las más cercanas al propósito de TreeDoc. Las *wikis* son gestores de contenidos orientados al desarrollo colaborativo. En ellas, los usuarios vuelcan su contenido, pero también actúan como lectores y correctores de otros contenidos. Las características de TreeDoc hacen que pueda utilizarse en este sentido, ya que se trata de una herramienta para gestionar los documentos y permite su edición en línea por parte de los usuarios.

La aparición de las aplicaciones web 2.0 es posible gracias a la evolución de las aplicaciones web hacia un modelo que permite una completa interacción con el usuario, que puede acceder a través del navegador web a aplicaciones completamente interactivas, las aplicaciones RIA (*Rich Internet Application*). El hecho de que se acceda a la aplicación a través del navegador permite que dispositivos muy diferentes puedan conectarse entre sí, y que no sea necesaria la instalación del sistema en el dispositivo. De este modo, podemos acceder a la misma aplicación desde nuestro ordenador (independientemente del sistema operativo) o con un dispositivo móvil con el único requisito de un navegador web.

Las tendencias anteriores confluyen en los editores colaborativos como TreeDoc. A fin de establecer las características que debería considerar TreeDoc se ha realizado una comparativa de varios de estos sistemas. El resultado se recoge en la tabla 1. Las características estudiadas son las siguientes:

- Web: Indica si el editor se puede utilizar en el navegador web o si por el contrario es una aplicación de escritorio.
- Gestión de documentos: Indica si el sistema proporciona mecanismos para la gestión de documentos
- Exportación a otros formatos.
- Colaboración: Si se pueden crear documentos de forma colaborativa entre varios usuarios. El apartado "En tiempo real" indica si esta colaboración puede ser editando el documento de forma simultánea.
- Contenido multimedia. Indica si el editor sólo soporta texto o también contenido multimedia adicional como imágenes, audio y video.
- Otras características: algunas características interesantes no contempladas en los campos anteriores.

Sistema	Web	Gestión de documentos	Exportación a otros formatos	Colaboración		Contenido multimedia	Otras características
					En tiempo real		
Word de Ms. Office	Sí	No	Sí	No hasta la versión 2011	No	Sí, imágenes, fórmulas, tablas.	Hasta la versión de 2011 cumple estas características; la última versión es más completa
Gobby	No	No	No	Sí	Sí	Sólo texto	Incorpora un chat para que los usuarios se comuniquen
AbiWord	No (aunque hay un plugin)	No	No	Sí	Sí	Sí, imágenes y tablas	Se le puede incorporar el plugin AbiCollab
Writely	Sí	No	Sí	No	No	Sólo texto	Permite asignar permisos a usuarios
Synchroedit	Sí	No	No	Sí	Sí	Sólo texto	La colaboración no es en tiempo real si el documento es de gran tamaño
Mobwrite	Sí	No	No	Sí	Sí	Sólo texto	La colaboración no es en tiempo real si el documento es de gran tamaño
Etherpad	Sí	No	No	Sí	Sí	Sólo texto	Uno de los primeros con colaboración en tiempo real, carácter a carácter
Google Docs	Sí	Si	Sí	Sí	Sí	Sí, imágenes, tablas, ecuaciones, fórmulas	El principal fallo es a la hora de insertar títulos y subapartados, y el formato de los documentos.

Tabla 1 "Comparación de sistemas"

La revisión de la Tabla 1 hace referencia a las características de la distribución estándar de los productos indicados. Estas pueden extenderse frecuentemente con plugins adicionales. Por

ejemplo, el plugin *Google Cloud Connect* para Microsoft Office lanzado en febrero de 2011 permite sincronizar los documentos de Office con *Google Docs* para editarlos a través de la nube.

En el Apéndice A: Sistemas de edición de texto se puede encontrar más información sobre los sistemas descritos.

Después de analizar los sistemas existentes para la gestión y edición de documentos de forma colaborativa, se pueden extraer varias conclusiones. En primer lugar hay que destacar que han experimentado un gran desarrollo recientemente, que se ha visto favorecido por las últimas tendencias tecnológicas. Los usuarios han ido adaptándose y migrando a las aplicaciones web. Las herramientas de escritorio más utilizadas han tenido que adaptarse para permitir la colaboración y edición en línea para seguir teniendo un lugar en el mercado. Por su parte, las aplicaciones web enfocadas a la edición y colaboración en línea han ido mejorando y añadiendo nuevas funcionalidades, convirtiéndose en potentes herramientas de gestión y edición de documentos. La posibilidad de colaboración en tiempo real ha sido una de las características que más ha evolucionado. Algunos editores incorporan los cambios tras un intervalo de tiempo de algunos segundos, mientras otros lo hacen cada intervalos muy cortos o cada vez que se introduce o modifica un carácter.

5 Características del sistema desarrollado

A la vista del análisis del estado del arte realizado en la sección 0, se realizó un catálogo de requisitos que debía ofrecer un editor colaborativo. Estos requisitos perseguían cubrir una funcionalidad básica constituida por: la posibilidad de editar y gestionar los documentos a través de una aplicación web; que estas tareas puedan ser realizadas de forma simultánea y colaborativa por varios autores. Esta sección describe las características de TreeDoc asociadas a dichos requisitos.

5.1 Aplicación Web

El sistema se ha implementado como una aplicación Web. La principal ventaja que proporciona es que hace que sea posible trabajar desde cualquier ordenador con conexión a Internet (terminal). De esta forma no se imponen unos requisitos elevados en cuanto a potencia de los terminales y la aplicación puede ser utilizada en cualquier terminal con navegador, independientemente de su sistema operativo.

También desaparece el problema de distribución de nuevas versiones del software. Un problema que siempre está presente en las tradicionales aplicaciones de escritorio. Siempre que aparece una nueva versión tiene que ser distribuida de alguna manera a los usuarios. Este es un tema importante, puesto que en muchos casos las nuevas versiones incorporan reparaciones de errores de la versión anterior. Si las nuevas versiones no son distribuidas adecuadamente, no solucionan el error en todos los equipos que tienen la aplicación instalada y provocan incompatibilidades.

5.2 Creación y edición de documentos

El sistema desarrollado consta con una aplicación para la creación de documentos, o la edición de documentos existentes. Aunque las herramientas que proporciona el editor permiten la creación de documentos con diferentes elementos, el formato del documento será uniforme.

Un documento TreeDoc está dividido en partes que pueden ser de diferentes tipos. Estos tipos son:

- Títulos: Los títulos sirven para delimitar secciones. Se pueden anidar unos títulos dentro de otros y así dotar de estructura al documento.
- Bloques de texto: Como su nombre indica, son áreas del documento formadas por texto.

- Fórmulas matemáticas: Se trabaja con ellas mediante la API de *Google Charts.* Se deben introducir en formato LaTex y el editor se encarga de mostrarlas en un formato más amigable para el usuario. Las fórmulas tienen asociado un pie editable, como puede verse en la figura.

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Ilustración 5-1 Fórmula

- Imágenes: Es posible insertar una imagen usando la URL de su localización o bien subiendo un archivo local. Se puede asociar un pie a una imagen.
- Vídeos: Los documentos pueden incluir vídeos. Los vídeos se mostrarán en un reproductor donde el usuario previsualizar el contenido del vídeo seleccionado para incorporarlo en el documento. Los vídeos a incluir deberán estar disponibles en Youtube. Con esta elección se ha buscado proporcionar al usuario un mecanismo para insertar contenidos que en los editores tradicionales no era posible. Aunque los documentos de TreeDoc se pueden imprimir, está orientado a documentos digitales, y por tanto ofrece la posibilidad de insertar tipos de partes que pueden ser visualizados en el navegador. Si el documento se va a ver de forma estática, se visualizará una imagen estática del vídeo insertado. Los videos tienen asociados pie editables para suministrar una descripción breve.
- Enlace a otro documento: Los enlaces permiten insertar en un documento una parte de otro documento sin copiarlo. Cuando se vincula una parte de un documento A en otro documento B, la parte de B se puede ver en A. En el caso en que se modifique la parte vinculada en B, estos cambios se reflejarán en A. Esta vinculación se realiza de forma transitiva. Es decir, si la parte vinculada posee a su vez un vínculo a otra parte, todas las partes se visualizarán en el documento que realiza el primer vínculo. La inclusión de un mecanismo de este tipo permite relacionar documentos, y mantener sincronizada la información incluida de un documento externo. De esta forma se evitan errores a la hora de copiar y además se garantiza que los dos documentos relacionados sean consistentes.
- Comentarios: Un comentario es una nota sobre una parte concreta de un documento. Se trata de información relacionada con una parte del documento pero que no forma parte del documento. Los comentarios se pueden emplear como herramienta de comunicación entre usuarios y de ayuda en la edición. Por ejemplo, un usuario puede incluir comentarios sobre cambios que debe realizar en el futuro sobre un documento, o sobre partes incompletas.

5.3 Sangrado automático

La aplicación sangra automáticamente el texto en función del nivel en el que se encuentra. Es decir, si se inserta un título para un apartado, el texto que va dentro del apartado se sitúa con mayor margen. También se generan automáticamente números para los títulos y subtítulos de la siguiente forma. El esquema resultante para el texto es:

1. Título de nivel 1

    Texto que va dentro del apartado de Título 1.

2. Título 2 de nivel 1

    2.1 Título de nivel 2 dentro de Título 2

        Texto del apartado 2.1

    2.2 Título de nivel 2 dentro de título 2

5.4 Colaboración en tiempo real

El editor desarrollado permite que varios usuarios puedan editar el mismo documento simultáneamente. La edición simultánea se puede realizar a nivel de documento, no a nivel de parte de un documento. Cuando un usuario está editando una parte de un documento, esta parte queda bloqueada para el resto de usuarios impidiendo que lo editen a la vez. El resto de usuarios son avisados del bloqueo mediante un pequeño icono en la zona superior derecha de cada parte.

5.5 Traducción colaborativa de documentos

Cuando se crea una parte de un documento, se asocia a la misma el idioma en el cual se encuentra escrita. Por defecto, se asigna el idioma de la sesión, aunque el usuario puede seleccionar otro. Es posible crear traducciones de una parte dada en otros idiomas, lo que permite visualizar un documento en distintos idiomas.

5.6 Multi-idioma

Los usuarios tienen vinculado un idioma por defecto en su perfil, que es guardado cuando se realiza el registro del usuario. El usuario puede modificar este idioma. Este idioma determina el idioma por defecto de la sesión del usuario en TreeDoc.

A la hora de visualizar un documento, cada parte se muestra en el idioma preferido por el usuario y que esté disponible para esa parte. En primer lugar se busca la parte en el idioma de la sesión (que es el idioma por defecto del usuario a no ser que éste lo haya cambiado). Si no está disponible en este idioma, se busca en otros idiomas, según el orden de preferencia del usuario.

5.7 *Check-in* y *check-out* de partes de documentos

El sistema cuenta con procedimientos que permiten que un usuario trabaje en un documento con la seguridad de que no sobrescribe o borra cambios que ha realizado otro usuario. Cuando un usuario está modificando una parte, automáticamente se realiza el *check-in* (bloqueo) de ésta. Cuando el usuario termina de editar una parte, se guardan los cambios y se lleva a cabo el *check-out* (desbloqueo) de la parte, permitiendo que el resto de usuarios pueda trabajar en ella.

5.8 Gestión de usuarios

Es posible registrar nuevos usuarios en la aplicación. Cuando se registra un usuario, se almacenan su nombre, dirección de correo electrónico y contraseña. Cada usuario puede acceder a sus datos y modificarlos en cualquier momento.

Los usuarios pueden también agregar otros usuarios a su lista de contactos. De esta manera puede acceder rápidamente a información útil del contacto y escribir notas sobre él.

La lista de contactos resulta útil para que el usuario lleve un control de otros usuarios con los que colabora, y pueda almacenar información general sobre ellos, no sólo información relacionada con un documento concreto como ocurre en el caso de los comentarios.

5.9 Protección de la sesión mediante *password*

Los usuarios deben estar registrados previamente en el sistema para poder acceder. Una vez registrados, acceden introduciendo su dirección de correo electrónico y contraseña.

A fin de evitar malos usos, las sesiones están protegidas por un mecanismo de temporización. Si un usuario no tiene actividad durante un tiempo determinado, su sesión se cierra automáticamente.

5.10 Varios documentos a la vez

El sistema permite a un usuario trabajar simultáneamente en más de un documento. Para ello necesitará abrir diferentes ventanas o pestañas del navegador Web.

5.11 Control de acceso y control de contenido

El sistema permite asignar permisos a los documentos para controlar el acceso a los mismos. Estos permisos son de lectura y escritura y se establecen para cada documento a nivel de usuario. Un usuario con permiso de lectura puede visualizar un documento, pero no puede modificarlo. Un usuario con permiso de escritura puede visualizar el documento y modificarlo. Si no se poseen permisos para un documento, éste no se puede visualizar.

El usuario que crea el documento tiene todos los permisos sobre el mismo. Por defecto, el resto de usuarios no tienen ningún permiso sobre ese documento, por lo que no pueden visualizarlo. El usuario que crea el documento puede asignar permisos de lectura y/o escritura al resto de usuarios. Los usuarios que cuentan con permiso de escritura sobre un documento también pueden asignar permisos a otros usuarios sobre ese documento.

5.12 Registro de actividad

Cada acción (crear un documento, editar una parte, insertar un mapa, etc.) queda registrada en el sistema almacenando el autor, el instante de tiempo y el tipo de acción realizada. De esta forma se pueden crear estadísticas que sirven para evaluar a los usuarios, detectar actividades vandálicas, etc.

5.13 Historial de evolución y control de versiones

Se incorpora un control automático de versiones sobre los documentos basado en el registro de actividad. Esto significa que se almacenan automáticamente todas las versiones de cada documento. Una versión incluye todos y cada uno de los cambios que se producen en un documento (exceptuando los permisos), y cierta meta-información como el autor y fecha de los cambios. Esto permite recuperar una versión anterior de un documento dada una fecha concreta. Para facilitar la búsqueda de instantes importantes en la vida del documento, es posible introducir marcas de tiempo llamadas "*publicaciones*" que facilitan la búsqueda de

versiones a los usuarios. Cuando se crea una publicación, el usuario puede introducir un nombre para la misma y así facilitar su posterior identificación.

El siguiente ejemplo muestra cómo funciona el control de versiones:

> "El documento Dispositivos de computadores se comenzó a escribir en 1970 y ya ha tenido varios cambios importantes dado el vertiginoso avance de la técnica. En el documento actual se encuentran capítulos dedicados al Blue Ray pero ninguno sobre los Floppy Disc. Sin embargo, podemos acceder a la versión del 1 de enero de 1990 del mismo documento para poder leer el capítulo sobre los Floppy Disc."

5.14 Percepción de la actividad de otros usuarios

Un aspecto importante de los entornos colaborativos es cómo percibe un usuario la actividad que está siendo llevada a cabo por el resto de usuarios. El término para referirnos a esta percepción de lo que llevan a cabo los demás usuarios es *awareness*. TreeDoc usa dos mecanismos para proporcionar *awareness*: comentarios y notificaciones de estado.

Se puede utilizar una herramienta de escritura de comentarios para que un usuario escriba una explicación del trabajo que está llevando a cabo. El resto de usuarios pueden leer los comentarios, y de este modo saber lo que están haciendo otros. Los comentarios sólo están disponibles en modo de edición y no forman parte del contenido del documento. Por este motivo, cuando se imprime o exporta el documento, los comentarios no se incluyen.

Las notificaciones de estado están relacionadas con indicaciones de si un bloque está o no siendo editado. Cuando un usuario comienza a editar una parte libre (sin bloquear), ésta se bloquea automáticamente para el resto de usuarios, que visualizan un pequeño icono informativo en la zona superior derecha.

5.15 Exportación a otros formatos

La aplicación permite la exportación de documentos a distintos tipos de formato, la mayoría de ellos libres. Los formatos a los que se puede exportar un documento son los siguientes:

- *HTML (HyperText Markup Language* -Lenguaje de Marcado de Hipertexto). Es el lenguaje de marcado más utilizado para crear páginas web. Proporciona un mecanismo de marcado para describir la estructura del documento y el contenido mediante etiquetas.

- *PDF (Portable Document Format*, formato de documento portátil). Este formato creado por Adobe Systems presenta el documento tal y como se imprimiría, sin que haya cambios posteriores de maquetación. Es uno de los formatos más usados para el intercambio de archivos.
- *TXT (Plain Text* - Texto Plano). El texto plano presenta el texto sin formato; únicamente los caracteres. Es útil contar con una exportación a este tipo de formato cuando queremos generar únicamente texto, sin tener en cuenta el tipo de letra ni aspectos de formato del documento.
- *TDF (TreeDoc Format* - Formato de TreeDoc). Es un formato que almacena los datos y la estructura del documento de la misma forma en la que representan en el editor. Está indicado para hacer copias de seguridad de la última versión del documento a nivel personal. No almacena ni el historial ni los usuarios que han intervenido en la edición.
- *ODT (OpenDocument*). OpenDocument es un estándar para documentos ofimáticos desarrollado por distintos organismos y empresas, visado por organismos de estandarización independientes, y que puede ser utilizado por cualquier proveedor sin pago de licencias. Está basado en un esquema *XML* inicialmente creado e implementado por la suite de aplicaciones informáticas OpenOffice.
- *EPUB* (*Electronic publication* - Publicación electrónica). Formato de código abierto para archivos de libro electrónico (*e-book*) creado por el *International Digital Publishing Forum* (IDPF). En el formato de libro digital ePub se marca el contenido, pero no se delimita su formato, de tal forma que se puede visualizar en las pantallas de los lectores de libros electrónicos aunque tengan diferente tamaño.

6 Diseño y modelado del sistema

6.1 Casos de uso

A partir de los requisitos se han identificado casos de uso relacionados con los documentos y con los usuarios. Los relacionados con los documentos pueden dividirse en dos grupos: los casos de uso relacionados con la gestión de los documentos y los relacionados con la edición de un documento concreto. Los casos de uso relacionados con los usuarios se clasifican también en dos grupos: control de sesiones y gestión de la información del usuario, dentro de la cual se incluye la información personal y los contactos del usuario.. A continuación se discuten estos casos con más detalle.

El control de sesión incluye los dos casos básicos que se muestran en la Ilustración 6-1. Es necesario iniciar sesión para poder realizar cualquier otra acción con TreeDoc. Una vez iniciada la sesión, también es posible salir de la aplicación (cerrar sesión).

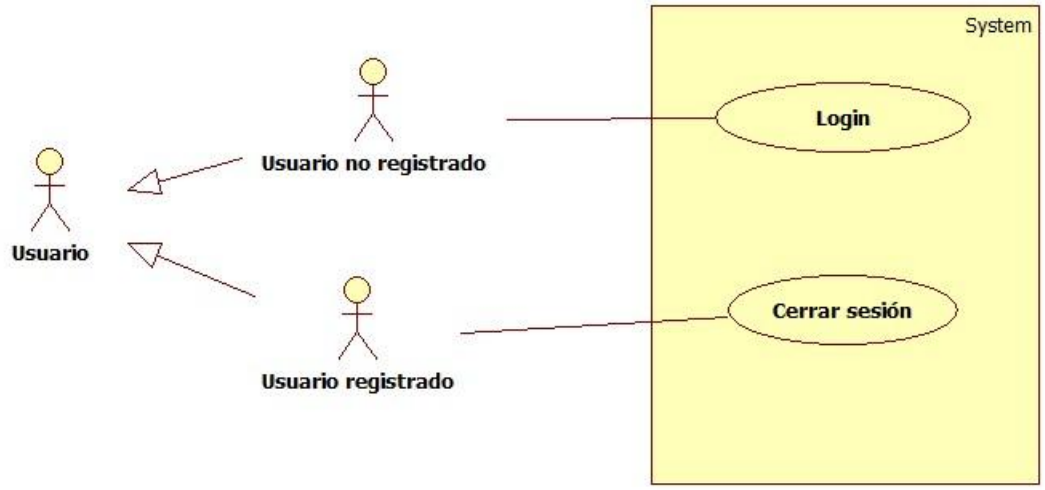

Ilustración 6-1 Casos de uso básicos: login y cerrar sesión

El caso de uso de *Login* permite que un usuario inicie sesión para acceder al sistema introduciendo su nombre de usuario y su contraseña. Un usuario que ya ha iniciado sesión se representa en los diagramas como “Usuario registrado”. En la Ilustración 6-1 también se muestra el caso de uso de Cerrar sesión, ya que un usuario puede cerrar sesión para salir del sistema.

Los siguientes casos de uso requieren que el usuario que los lleva a cabo sea un usuario que ha iniciado sesión, es decir, un “Usuario registrado”.

6.1.1 Casos de uso de documentos

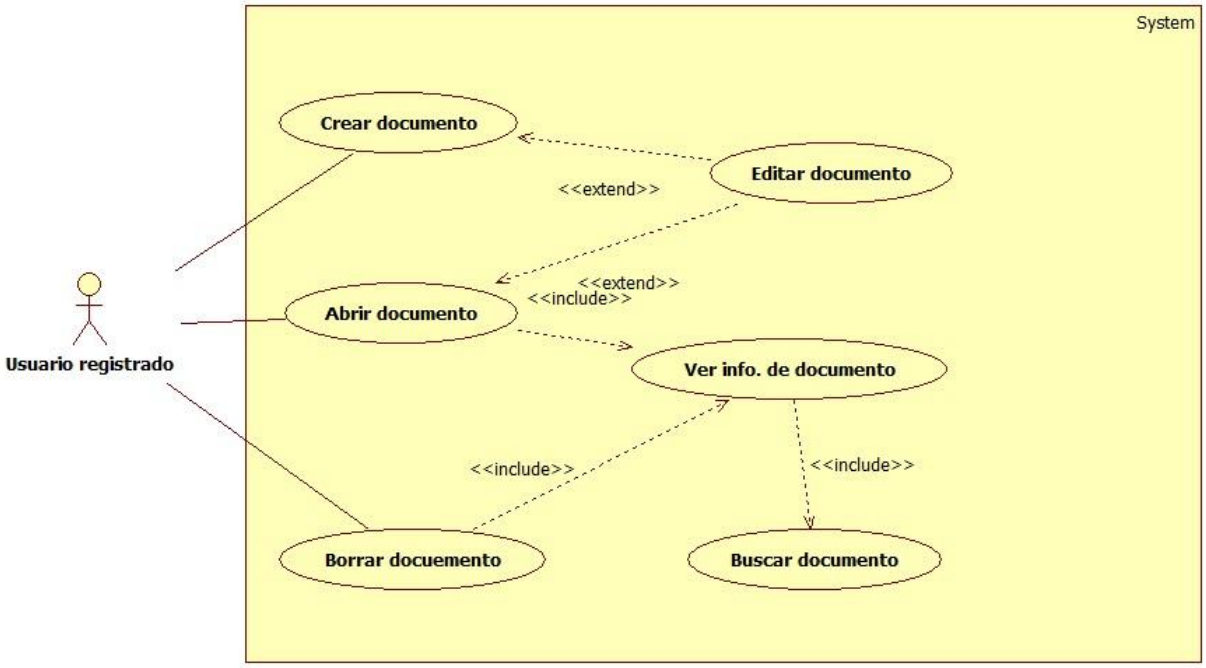

Ilustración 6-2 Casos de uso de gestión de documentos

La **¡Error! No se encuentra el origen de la referencia.**Ilustración 6-2 muestra un diagrama de casos de uso que representa los casos de uso relacionados con la gestión de documentos. Cuando un usuario ha iniciado sesión en el sistema, puede acceder a la parte de la aplicación de gestión de documentos, que le permite crear un documento nuevo, abrir un documento existente, o borrar un documento existente. Cuando se decida crear un documento nuevo, se presentará el editor de documentos y un documento vacío para comenzar con su edición. Para editar o borrar un documento será necesario seleccionar el documento que se desea editar/borrar. Para ello se proporciona un buscador de documentos, y esta acción aparece en el diagrama de la **¡Error! No se encuentra el origen de la referencia.**Ilustración 6-2 en el caso de uso "*Buscar documento*". Una vez que se encuentra el documento deseado, se accede a la información del mismo (caso de uso *Ver info. de documento* del diagrama de casos de uso mencionado), que incluye:

- Título
- Descripción
- Palabras clave
- Autor
- Fecha de modificación

Una vez que se ha mostrado la información del documento, será posible abrirlo para su edición (Caso de uso *Abrir documento* del diagrama) o borrarlo (Caso de uso *Borrar documento*).

Tanto para crear un documento nuevo como para editar un documento existente, se utilizará el editor de documentos. Los casos de uso relacionados con el editor se muestran en el diagrama de la Ilustración 6-3.

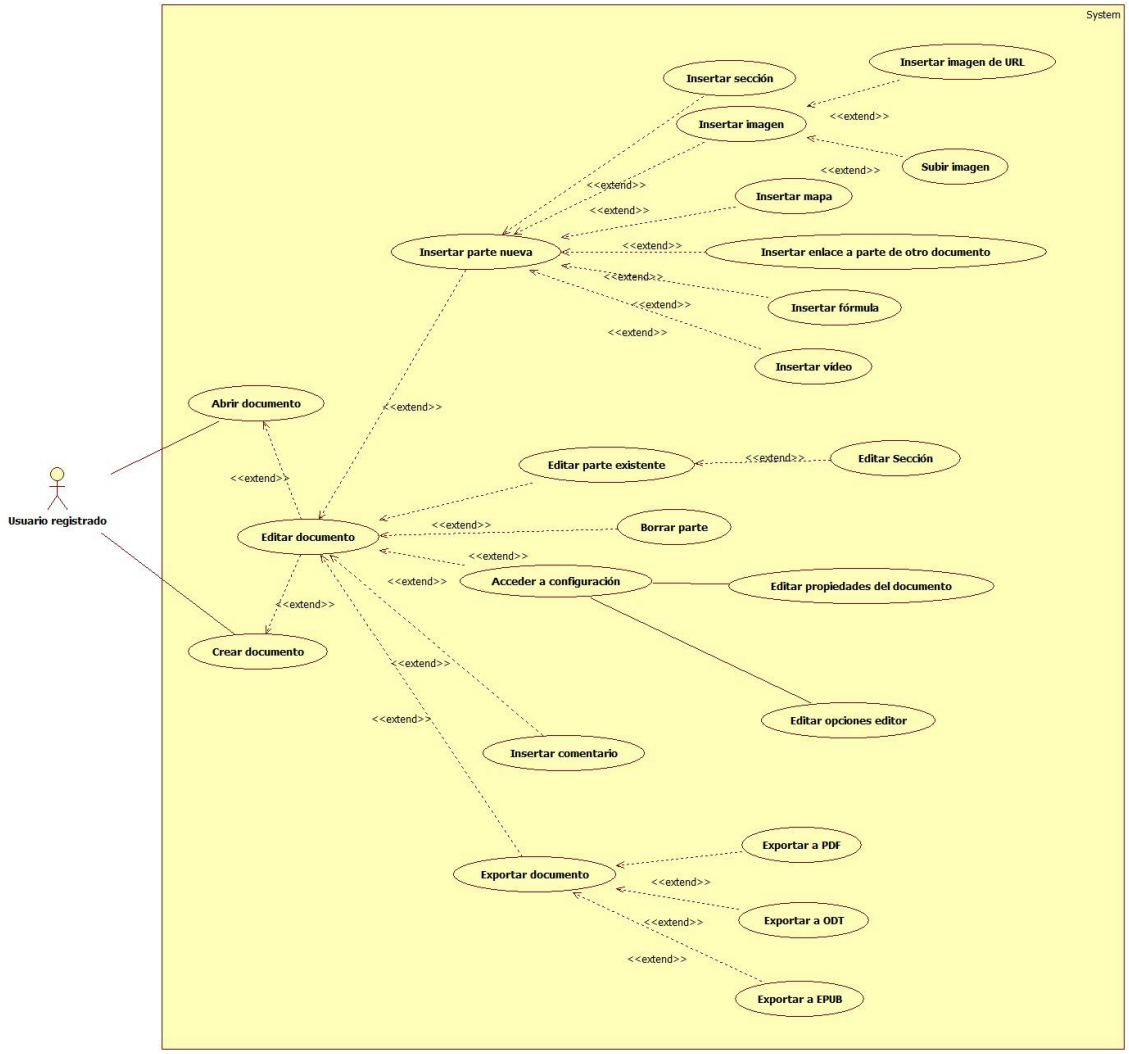

Ilustración 6-3 Casos de uso del editor de documentos

El diagrama de casos de uso anterior (Ilustración 6-3) muestra los casos de uso relacionados con Editar documento. Es decir, son los casos de uso relacionados con el editor. Para más información sobre las funciones del editor, consultar el apartado 5.2 "Creación y edición de documentos".

Es posible insertar una parte nueva; en ese caso, será necesario seleccionar un tipo de parte; esto se refleja en el diagrama con los casos de uso siguientes:

- *Insertar Sección*: Permite insertar una nueva sección. Cuando se inserta una sección se puede introducir el título y el texto de la misma.
- *Insertar imagen*: Permite insertar una imagen; hay dos formas de hacerlo:
    - *Insertar imagen de URL*: Permite introducir la *URL* de la imagen que se quiere insertar
    - *Subir imagen*: Permite seleccionar la ruta de la imagen que se quiere subir del disco duro local.
- *Insertar mapa*: Permite introducir un mapa de Google Maps (8).
- *Insertar enlace a otro documento*: Para insertar una parte de otro documento se proporciona un buscador de documentos. El diagrama muestra el caso de uso Buscar documento, que es el mismo que se muestra en laIlustración 6-2. Una vez seleccionado el documento deseado, se mostrará un esquema de las partes de ese documento y será posible seleccionar la parte que se desea enlazar (Caso de uso *Seleccionar parte del diagrama*).
- *Insertar fórmula*: Permite introducir una fórmula matemática. Es necesario introducirla en formato *LaTex* (9). Para facilitar la edición se ofrece la posibilidad de introducir algunos de los elementos más comunes con ayuda de botones. Los elementos que se proporcionan son:
    - Fracción
    - Raíz
    - +/-
    - Alfa
    - Beta
    - Gamma
    - Pi
    - Sumatorio
    - Límite
- Insertar vídeo: Se puede insertar un vídeo de *Youtube*(10). Para ello se proporciona un campo de texto donde el usuario puede introducir el título del vídeo que desea introducir.

También es posible editar una sección que ya ha sido creada del documento. El caso de uso *Editar parte existente* tiene el caso de uso relacionado de *Editar sección*. Se puede seleccionar la sección deseada y modificar el texto de la misma. Una parte existente también se puede borrar tras seleccionarla (Caso de uso *Borrar parte*).

El caso de uso *Acceder a configuración* tiene dos casos de uso directamente relacionados:

- *Editar propiedades del documento*. Las propiedades del documento que pueden editarse son:
    - *Nombre del documento*. Se puede modificar el texto del nombre del documento. Los documentos tendrán un nombre por defecto hasta que sea editado. Concretamente se crean con el nombre “Nuevo documento”.
    - *Descripción*. Es posible introducir una descripción del documento.
    - *Keywords.* En el campo Keywords se pueden introducir palabras clave del documento.
- *Editar propiedades del editor*.

El editor también incluye una opción para insertar un comentario en el documento. Aparecerá el nombre del usuario que inserta el comentario y el texto.

El caso de uso *Exportar documento* muestra los casos de uso relacionados que definen los formatos de exportación posible; por esto aparecen los casos de uso de Exportar a PDF, ODT y EPUB.

6.1.2 Casos de uso de contactos

La gestión de contactos se presenta en el siguiente diagrama:

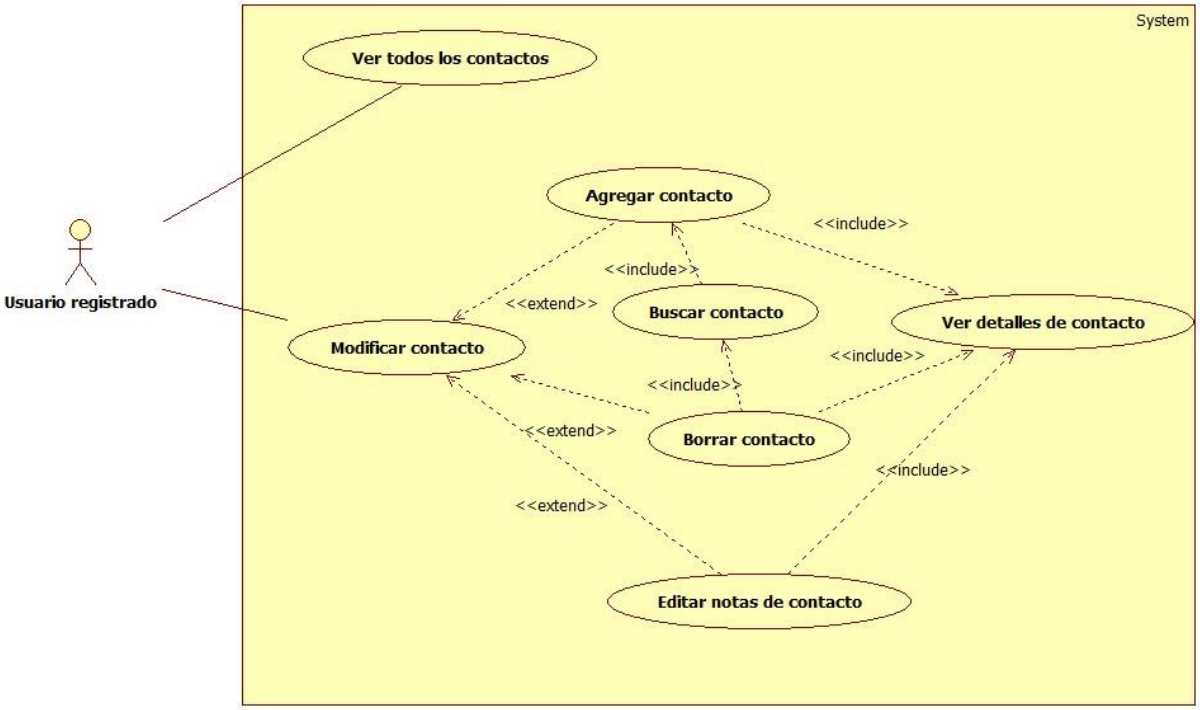

Ilustración 6-4 Casos de uso de gestión de contactos

Los casos de uso relacionados con la gestión de contactos se muestran en la Ilustración 6-5.

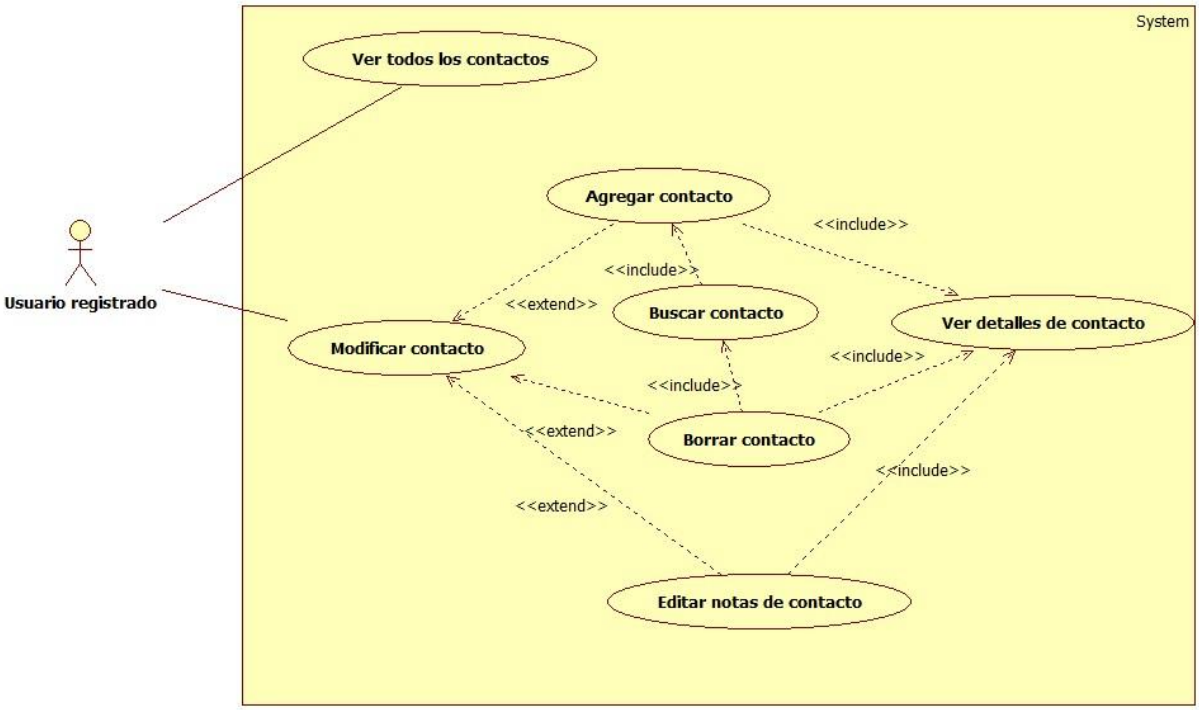

Ilustración 6-5 Casos de uso de gestión de contactos

Es posible acceder a un listado de todos los contactos, y utilizar un buscador para localizar a un contacto en concreto.

Para modificar un contacto hay que seleccionarlo primero (caso de uso *Modificar contacto* del diagrama de la Ilustración 6-5). Después se podrán consultar los detalles de ese contacto (Caso de uso *Ver detalles de contacto*) y realizar las siguientes acciones sobre el contacto:

- *Editar notas de contacto*. Se podrán escribir notas sobre el contacto que sólo el usuario puede consultar.
- *Agregar contacto* (si no forma parte de los contactos del usuario) y *Borrar contacto* si ya forma parte de sus contactos.

El sistema también debe proporcionar funcionalidad para editar la información del usuario. El siguiente diagrama muestra los casos de uso relacionados con esta funcionalidad:

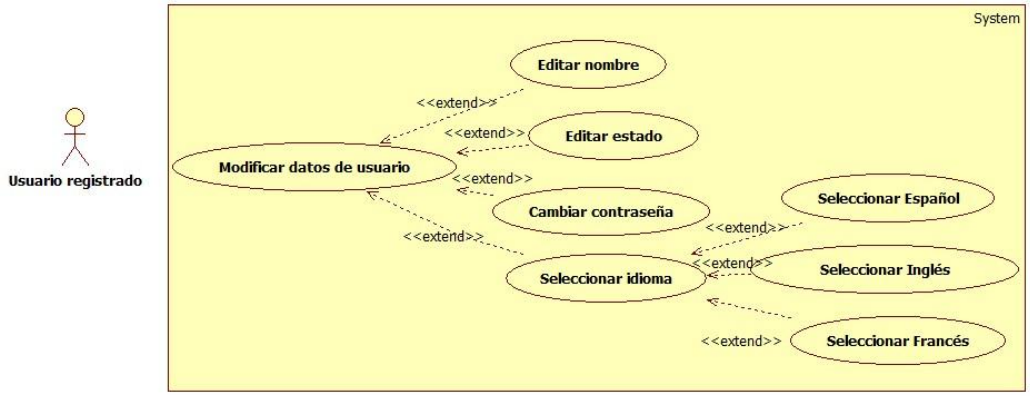

Ilustración 6-6 Casos de uso de modificación de información de usuario

La modificación de los datos de un usuario puede consistir en:

- *Editar nombre*: Editar el texto del nombre del usuario.
- *Editar estado*: Introducir un texto que describe el estado.
- *Cambiar contraseña*: Cambiar la contraseña con la que el usuario inicia sesión
- *Seleccionar idioma*: Se puede cambiar el idioma de la aplicación y definir un idioma diferente para los documentos (ver apartado

- Multi-idioma para más información sobre el idioma).

El sistema desarrollado a partir de los casos de uso que muestran los diagramas anteriores tiene secciones dedicadas a cada parte explicada en este apartado.

6.2 Interfaz gráfica

La interfaz gráfica se ha diseñado a partir de los casos de uso de la sección anterior de forma que la utilización del sistema sea lo más intuitiva posible. Para ello se han, utilizando disposiciones de los elementos y flujos de interacción similares a los que los usuarios pueden estar acostumbrados a utilizar.

A partir de los diagramas de casos de uso se ha dividido el sistema en varias partes diferenciadas:

6.2.1 Login

La página de login está formada por dos campos a rellenar por el usuario: el email del usuario y la contraseña:

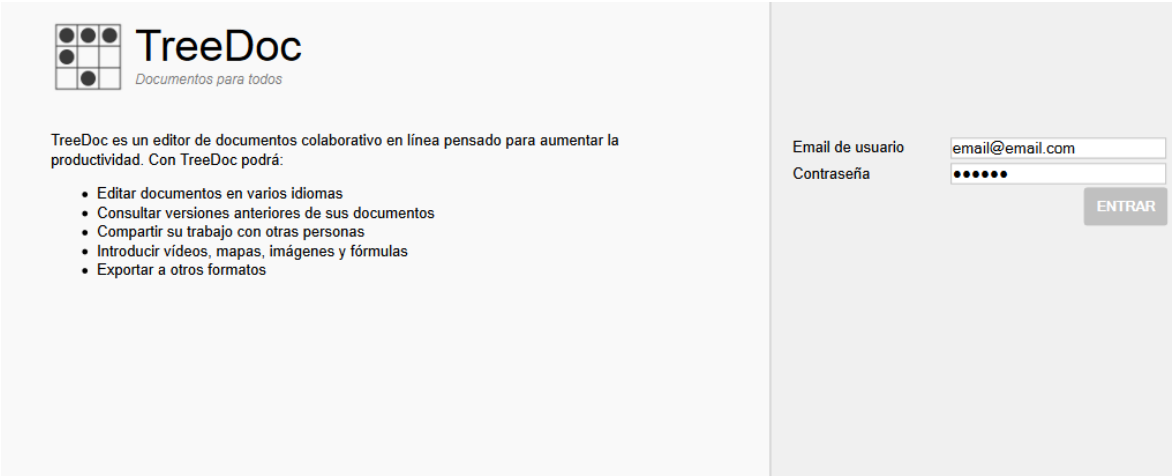

Ilustración 6-7 Cuadro de Login

Si no se introducen un nombre de usuario y contraseña válidos, se presentará la siguiente pantalla:

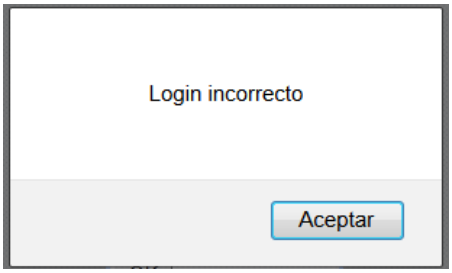

Ilustración 6-8 Login incorrecto

6.2.2 Barra de enlaces superior

La barra superior permite acceder a las distintas partes de la aplicación (documentos, contactos, configuración). También permite visualizar el nombre de usuario de la sesión actual, y salir de la sesión.

Ilustración 6-9 Barra de enlaces superior

6.2.3 Gestión de documentos

Una vez que se ha llevado a cabo el login, por defecto se accede a la parte de gestión de documentos de la aplicación.

Desde esta parte se pueden llevar a cabo las funciones descritas en los casos de uso de la**¡Error! No se encuentra el origen de la referencia.**Ilustración 6-2.

A continuación se muestra una imagen de esta pantalla y se explica cómo se han implementado los casos de uso para conseguir la funcionalidad deseada.

Ilustración 6-10 Pantalla de Gestión de documentos

Como se puede ver en la Ilustración 6-10, aparecen los siguientes elementos numerados:

1. Botón para crear un documento nuevo.
2. Buscador que permite buscar un documento por su título (Caso de uso *Buscar documento*).
3. Lista de documentos. Se muestra el resultado de la búsqueda de los documentos cuyo título contiene el texto introducido en el buscador. Si no se introduce ningún texto en el buscador, aparece el listado con todos los documentos.

Para obtener la información de un documento concreto, primero hay que localizarlo en la lista de documentos, a continuación aparecerá una pantalla similar a la que aparece en laIlustración 6-11.

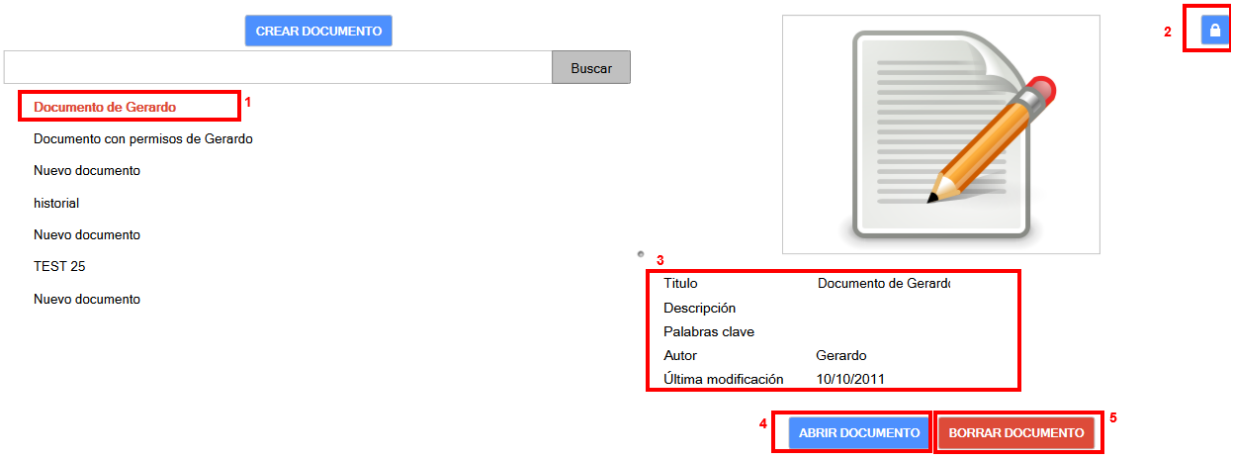

Ilustración 6-11 Pantalla de propiedades de un documento

Las áreas marcadas en la Ilustración 6-11 muestran las siguientes zonas:

1. Documento seleccionado en color rojo.
2. Botón de permisos: permite acceder a la pantalla de asignación de permisos al documento.
3. Propiedades del documento: Los campos *Título*, *Descripción* y *Palabras clave* son editables. *Autor* y *Última modificación* se rellenan automáticamente.
4. Botón para abrir el documento para su edición.
5. Botón para borrar el documento.

Los permisos sobre el documento se gestionan a través de una pantalla como la que aparece en la Ilustración 6-12. En esta pantalla se aprecia:

1. Buscador de usuarios por nombre; funciona de forma similar al buscador de documentos
2. Lista de usuarios que coinciden con la búsqueda. Cuando se selecciona un usuario se marca en rojo.

3. Botones Aceptar / Cancelar
4. Información del usuario: Nombre y e-mail.
5. Botones de permisos: es posible asignar permisos de lectura y/o escritura.

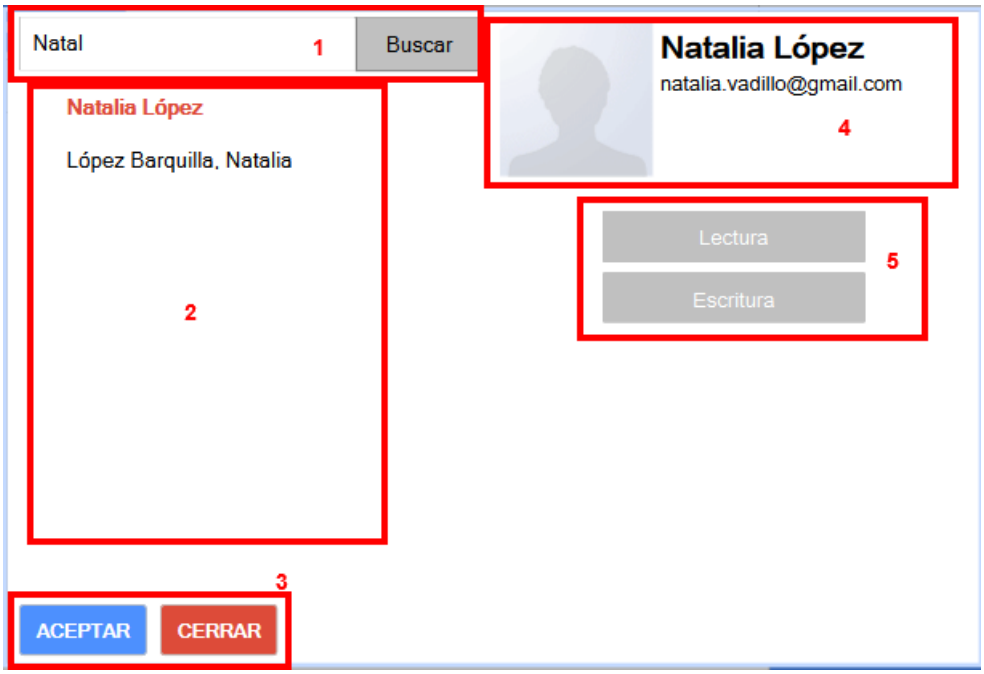

Ilustración 6-12 Pantalla de permisos

6.2.4 Gestión de contactos

La parte de gestión de contactos se muestra en la Ilustración 6-13. Por defecto se muestran los contactos que están en la agenda del usuario en la lista. Si se pulsa el botón “Todos”, se muestran todos los usuarios para poder acceder a su ficha y opcionalmente, añadirlo a nuestra lista de contactos. También aparece un buscador instantáneo que permite buscar un contacto por nombre.

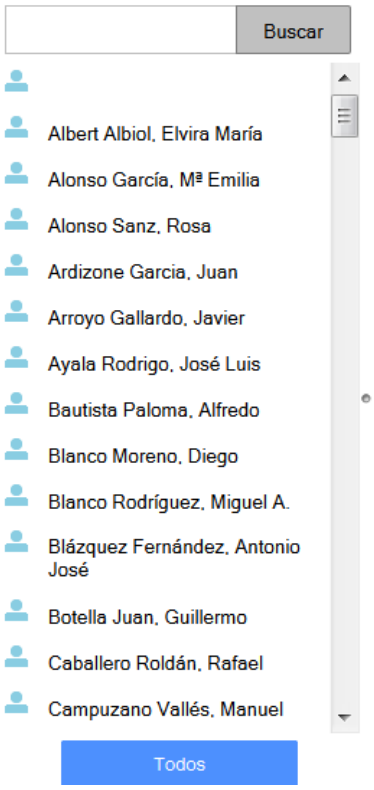

Ilustración 6-13 Pantalla de gestión de contactos

Seleccionando un contacto, se muestra una pantalla similar a la Ilustración 6-14.

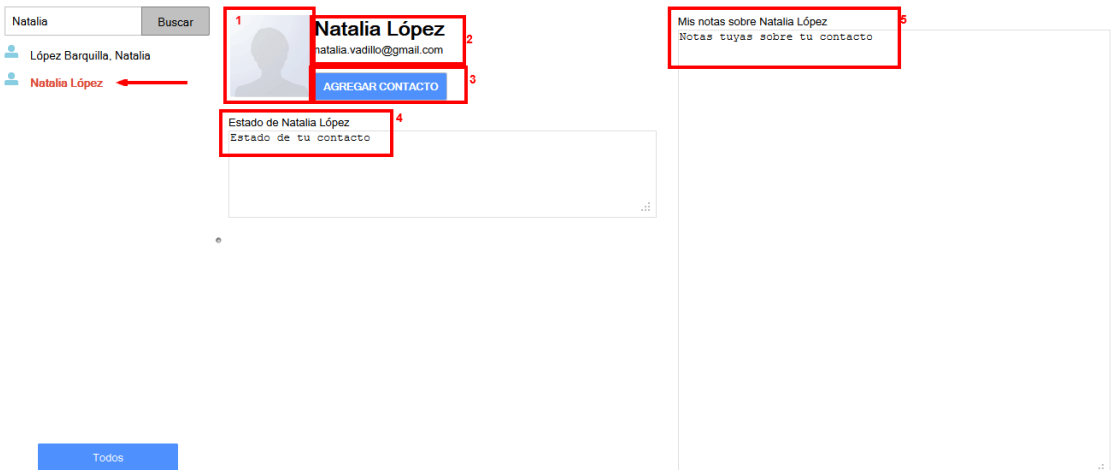

Ilustración 6-14 Pantalla de información de contacto

La imagen superior muestra en rojo las siguientes áreas de información:

1. Foto del contacto. Si no hay foto disponible se muestra una imagen por defecto.
2. Nombre y apellidos del contacto y dirección de correo electrónico.
3. Botón para agregar el contacto si no forma parte de los contactos del usuario. Si ya forma parte, el botón muestra la opción de “Borrar contacto”.
4. Estado del contacto.
5. Notas del contacto.

6.2.5 Editor del documento

Cuando se selecciona un documento para su edición, éste se muestra en el editor gráfico de documentos.

La pantalla principal del editor es la siguiente:

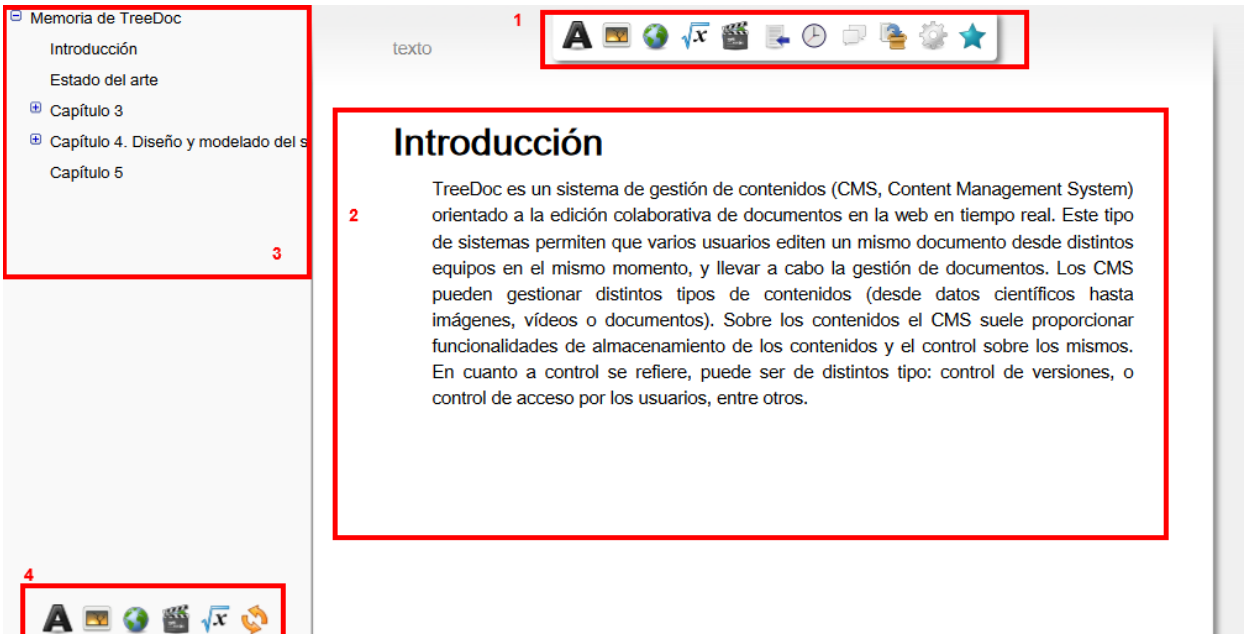

Ilustración 6-15 Pantalla de edición de documento

Áreas destacadas:

1. Barra superior Ofrece las opciones disponibles para añadir partes al documento.
   - Se puede insertar un nuevo título en el documento haciendo click en el icono  . Los títulos son como los que se muestran en el ejemplo de la Ilustración 6-15.
   - También puede insertarse una imagen haciendo click en el icono  . En este caso se mostrará un panel con las opciones posibles para insertar una imagen como el que aparece en la Ilustración 6-16.

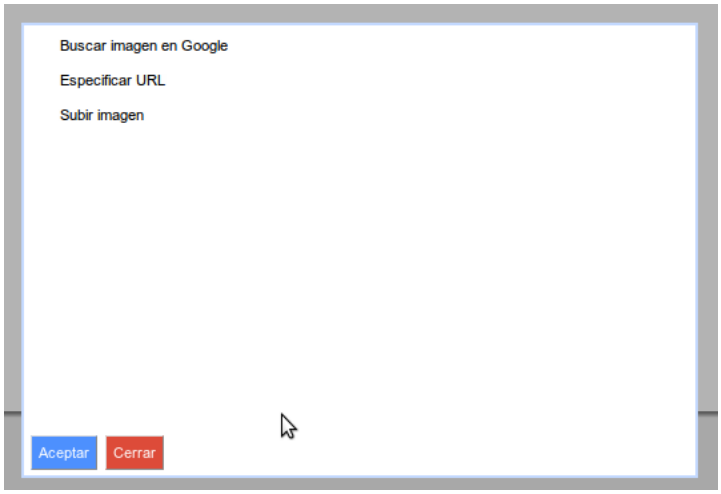

Ilustración 6-16 Pantalla de inserción de parte imagen en un documento

   - Se puede insertar una fórmula haciendo click en el botón  . Al seleccionar esta herramienta, se abre un editor para introducir el código en LaTex. La Ilustración 6-17 muestra un ejemplo del mismo.

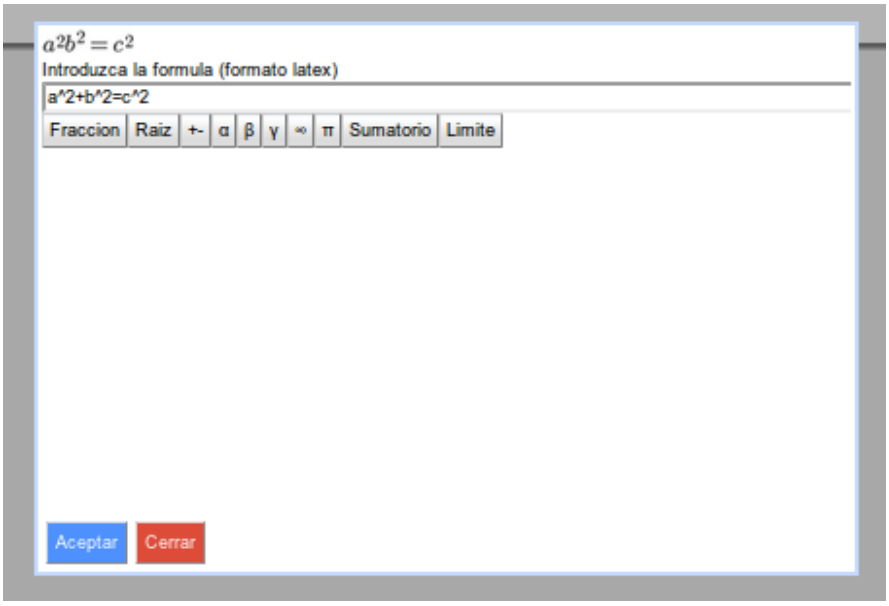

Ilustración 6-17 Pantalla de inserción de parte fórmula

- El icono  permite introducir un vídeo. Muestra una pantalla de este tipo para buscar vídeos de YouTube y seleccionar el deseado. La Ilustración 6-18 muestra la pantalla que aparece en este caso. Una vez que se selecciona un vídeo, se puede previsualizar antes de seleccionar “Aceptar”.

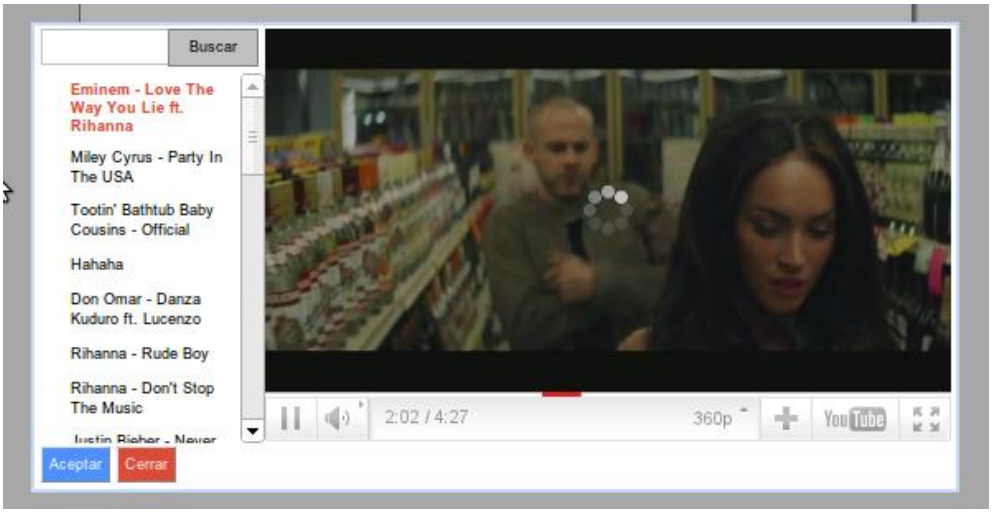

Ilustración 6-18 Pantalla de inserción de parte vídeo

- Se puede acceder al historial seleccionando  . Aparecerá una lista con las versiones anteriores del documento, y el usuario podrá seleccionar la versión que desea visualizar.
- El icono  permite insertar un mapa utilizando la *API* de *Google Maps.*
- Con el icono  se puede insertar un enlace a una parte de otro documento, seleccionando el documento y la parte que se desea enlazar. Estas acciones se muestran en la Ilustración 6-19.

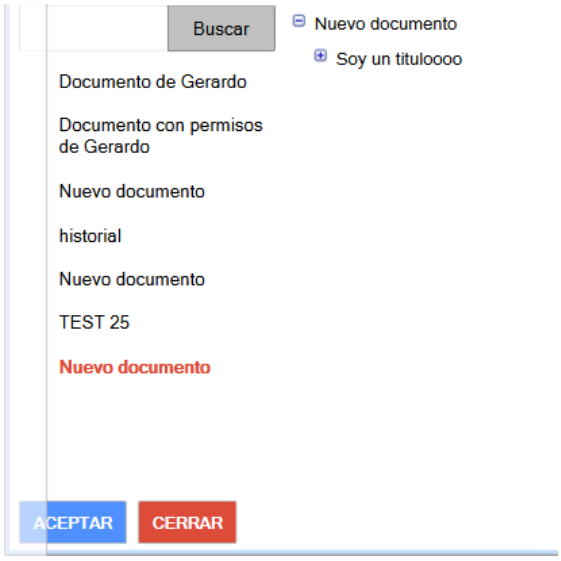

Ilustración 6-19 Pantalla de inserción de parte

- Para insertar un comentario en una parte del documento, hay que seleccionar  , y se podrá escribir el comentario. El resto de usuarios verán el nombre del usuario que ha introducido el comentario y el texto.
- Seleccionando  se puede exportar el documento a alguno de los formatos disponibles.
- El icono de  permite crear una publicación del documento. Cuando se selecciona, se debe introducir un nombre para la publicación.

- Con  se accede a la pantalla de la Ilustración 6-20, donde se pueden editar el título, descripción y palabras clave del documento, y acceder a la configuración del editor y seleccionar si mostrar o no el índice.

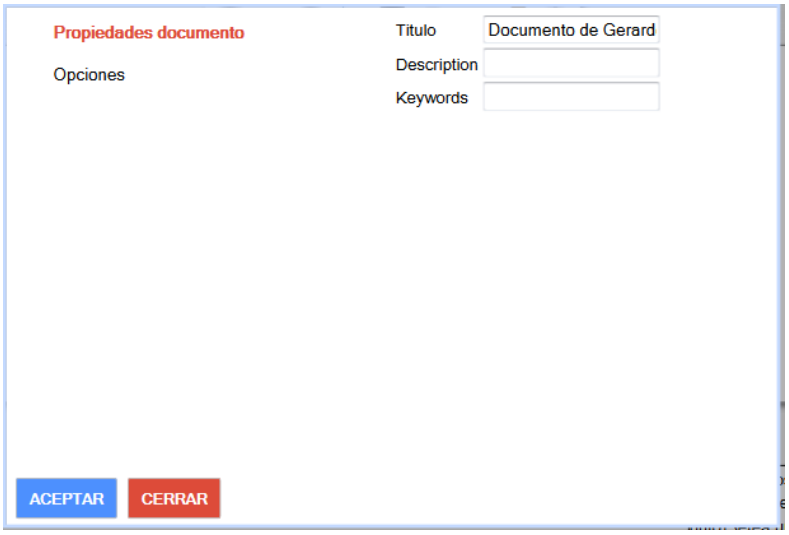

Ilustración 6-20 Pantalla de configuración de documento

2. Área de edición: En esta zona de la Ilustración 6-15 se muestra la representación del documento. Según se van insertando secciones, el sangrado se realiza de forma automática, y se permite agregar el título y el texto de la sección. Además, se muestran las imágenes, una captura del vídeo, la fórmula en formato final, y los mapas. Si se ha insertado un enlace a otro documento, aparece como si fuera una parte más del mismo documento.
3. Estructura del documento: Muestra un esquema con la estructura del documento que permite localizar los elementos del documento de una forma rápida. Esta área está relacionada con el área 4, que permite seleccionar qué tipos de elemento se desean mostrar.
4. Tipo de elemento a mostrar en el esquema de estructura. Esta barra permite seleccionar un tipo de elemento. La estructura del documento que se muestra en el área 3 mostrará un esquema con los elementos del tipo que se seleccione en esta barra.

6.2.6 Configuración

La opción de “Configuración” permite modificar la información de la cuenta del usuario. Aa continuación se muestra la Ilustración 6-21, una pantalla de ejemplo de esta parte.

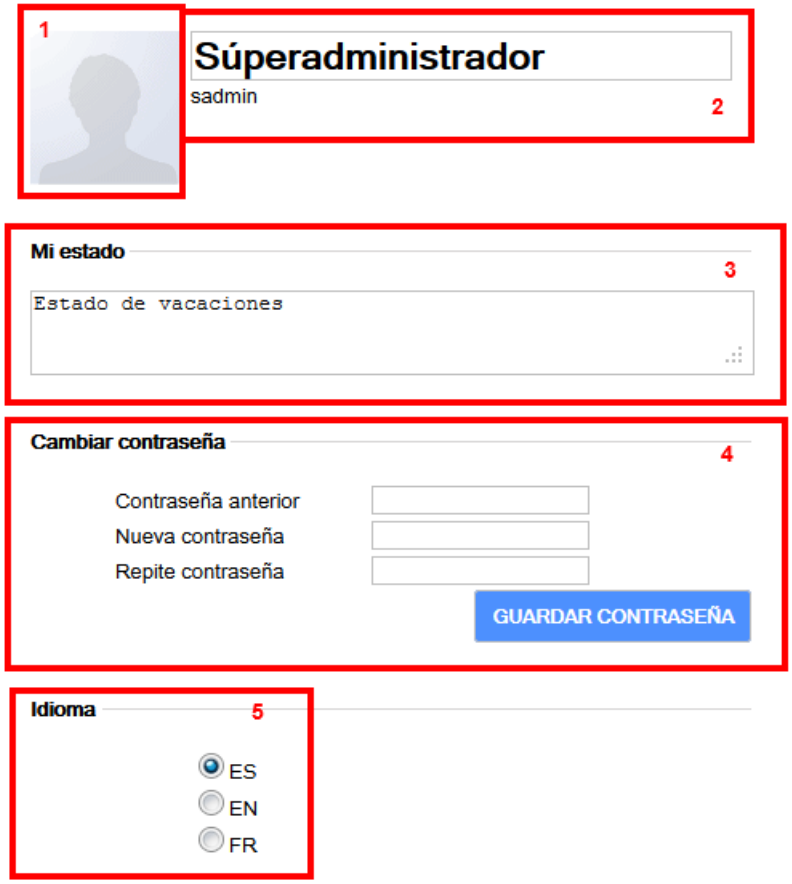

Ilustración 6-21 Pantalla de configuración

La pantalla incluye los siguientes elementos:

1. Foto. Si no se introduce una aparece una imagen por defecto.
2. Nombre completo y nombre que se muestra al resto de usuarios. Son campos editables que admiten entrada de tipo texto.
3. Estado. Permite escribir una descripción del estado que el resto de usuarios puede leer. Sirve para informar de estados como “De vacaciones” o “No disponible” por ejemplo.
4. Sección para cambiar la contraseña. Es necesario rellenar los 3 campos siguientes:
    a. Contraseña anterior: Contraseña que ya tiene asignada el usuario.

   b. Nueva contraseña: Contraseña que quiere establecer el usuario para iniciar sesión.
   c. Repite contraseña: Campo para introducir la nueva contraseña de nuevo por motivos de seguridad, para que no se produzcan errores al introducir la nueva contraseña por teclado.
   d. Después de introducir los 3 campos, hay que seleccionar el botón de "GUARDAR CONTRASEÑA"
5. Idioma: permite establecer el idioma por defecto del usuario entre las siguientes posibilidades:
   a. ES: Español
   b. EN: Inglés
   c. FR: Francés.

6.3 Diseño del sistema

Una vez explicado el funcionamiento básico de la aplicación se procede a explicar cómo está desarrollada. Para comenzar se explica la estructura de la aplicación mediante un diagrama de clases simplificado y sin atributos en las clases, en el que aparecen las relaciones entre los distintos objetos.

La arquitectura de la aplicación está basada en el patrón llamado modelo - vista - controlador (MVC) (11), con algunas variaciones sobre el MVC clásico. Este patrón separa tres niveles:

- Modelo: Representa la lógica de negocio de la aplicación, la información que se maneja.
- Vista: transforma el modelo en una página web que permite al usuario interactuar con ella.
- Controlador: Procesa las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

Un esquema que muestra a alto nivel esta separación está representado en la Ilustración 6-22.

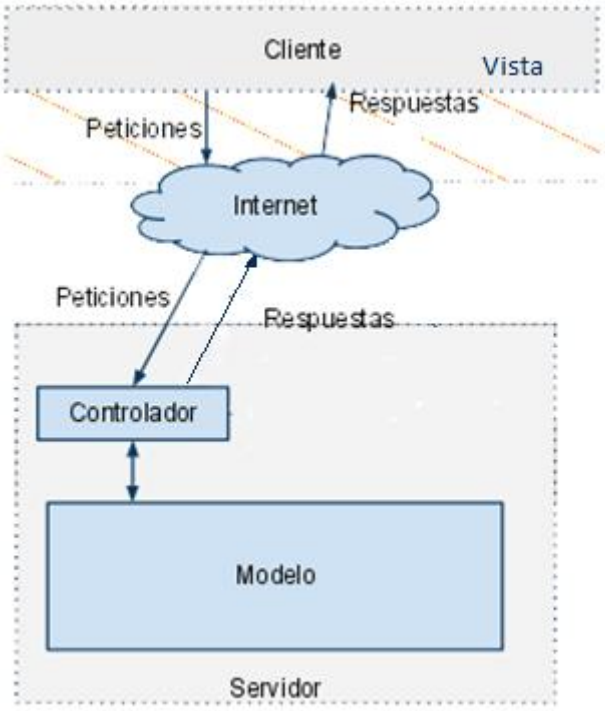

Ilustración 6-22 Arquitectura Modelo – Vista –Controlador.

El modelo a su vez está separado en una capa de abstracción de los datos y otra capa de acceso a los datos. De este modo se utilizan funciones para realizar consultas que no dependen de una base de datos concreta.

Desde el punto de vista funcional, los módulos principales del sistema se pueden ver en la Ilustración 6-23. Ésta muestra que la aplicación sigue el esquema cliente-servidor. Esto significa que el cliente hace peticiones al servidor; el servidor procesa estas peticiones, y envía la respuesta. Concretamente, es una aplicación Web, y por tanto, el cliente accede a ella a través de un navegador web. El modelo cliente-servidor se combina con el de MVC de tal forma que el modelo y el controlador residen en el servidor, y la vista se muestra en el cliente. Cuando el cliente envía una petición al servidor, el controlador es quien recibe la petición, la procesa (interactuando con el modelo), y envía la respuesta de nuevo a la vista, que se encuentra en el cliente.

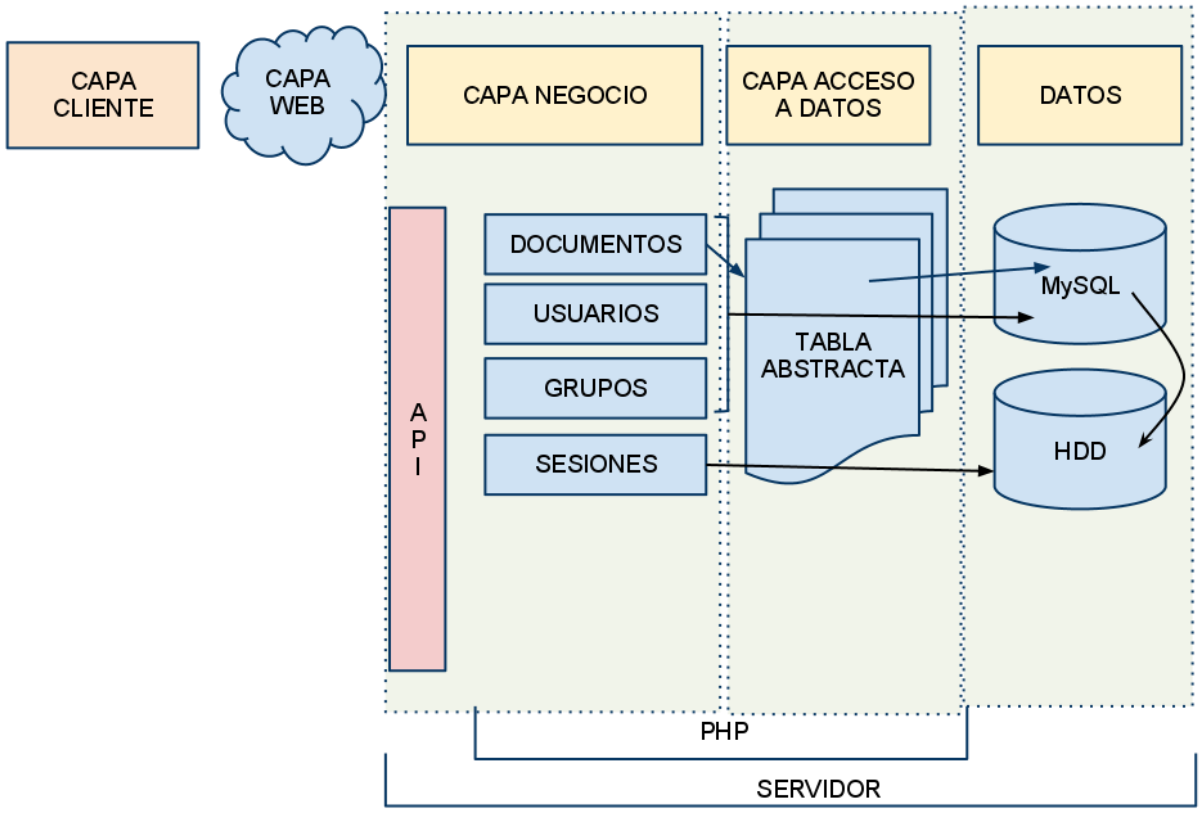

Ilustración 6-23 Esquema cliente-servidor de la arquitectura

6.4 Cliente

La aplicación del cliente está desarrollada utilizando el framework de Google GWT(12). Éste emplea varias tecnologías relacionadas con la web. Utiliza *CSS* para controlar la apariencia de la aplicación en cuanto a estilos y formatos. También utiliza la tecnología *AJAX* para llevar a cabo la comunicación asíncrona entre cliente y servidor, e incorpora fragmentos con JavaScript nativo. El desarrollo con Google GWT requiere herramientas como *Firebug* de *Firefox* o el *inspector DOM* de *Google Chrome* para depurar el código HTML que se presenta y el *JavaScript* generado por *GWT*. Ha sido necesario incorporar un log para ayudar en las tareas de depuración debido a que presentaban una gran dificultad.

6.4.1 Diagramas de clases de la interfaz gráfica

A continuación se presentan los diagramas de clases de la interfaz gráfica por módulos, y la relación que tienen estas clases con los elementos gráficos de la aplicación.

6.4.1.1 Módulo de contactos

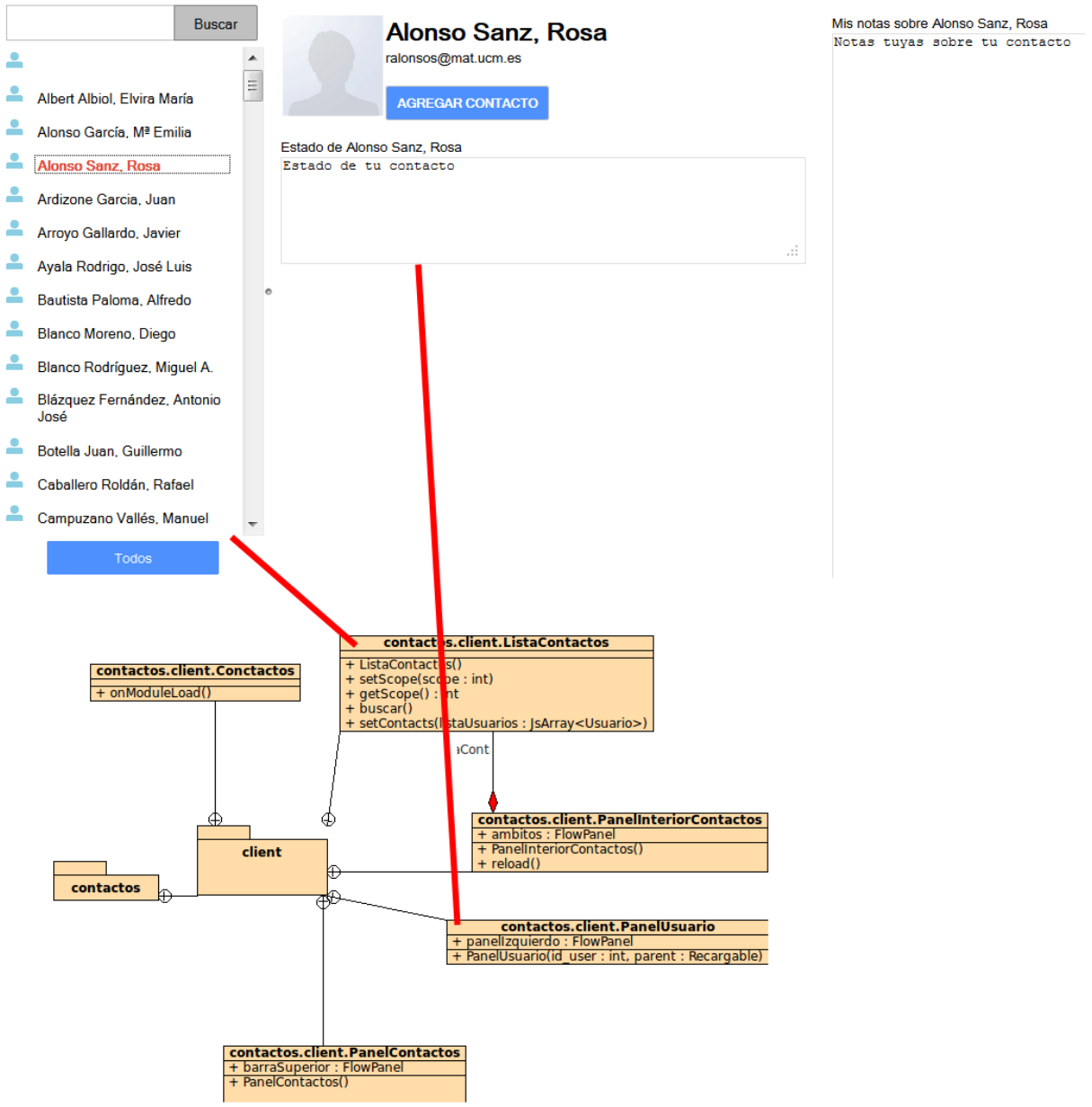

Ilustración 6-24 Módulo de contactos

6.4.1.2 Módulo configuración

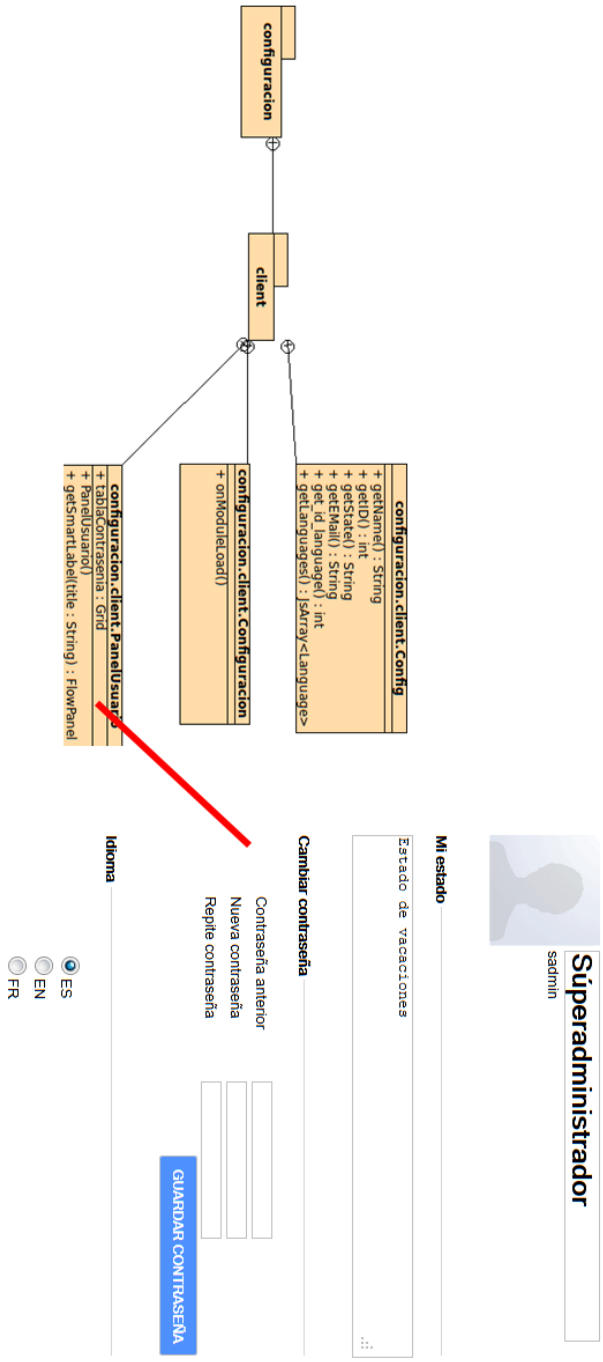

Ilustración 6-25 Módulo de configuración

El módulo de configuración está formado por un panel que contiene todos los elementos necesarios para seleccionar la configuración del usuario.

6.4.1.3 Módulo documentos

El módulo de documentos está formado por dos paneles principales que contienen el resto de componentes. El panel lateral izquierdo muestra el buscador y la lista de documentos, y el panel derecho muestra la información sobre el documento.

La Ilustración 6-26 también muestra la pantalla emergente que aparece cuando se selecciona el botón de permisos en el panel derecho.

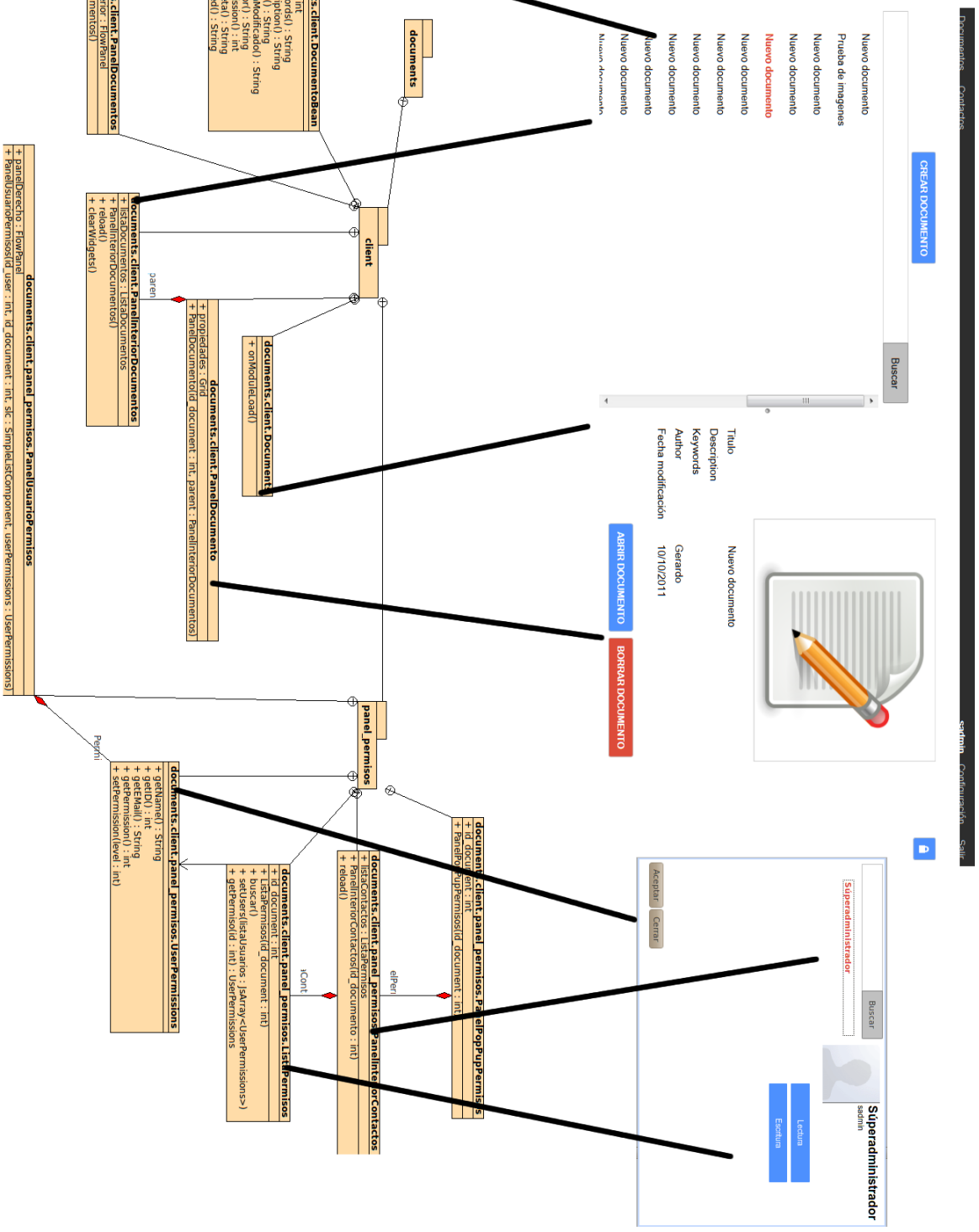

Ilustración 6-26 Módulo de documentos

6.4.1.4 Editor de documentos

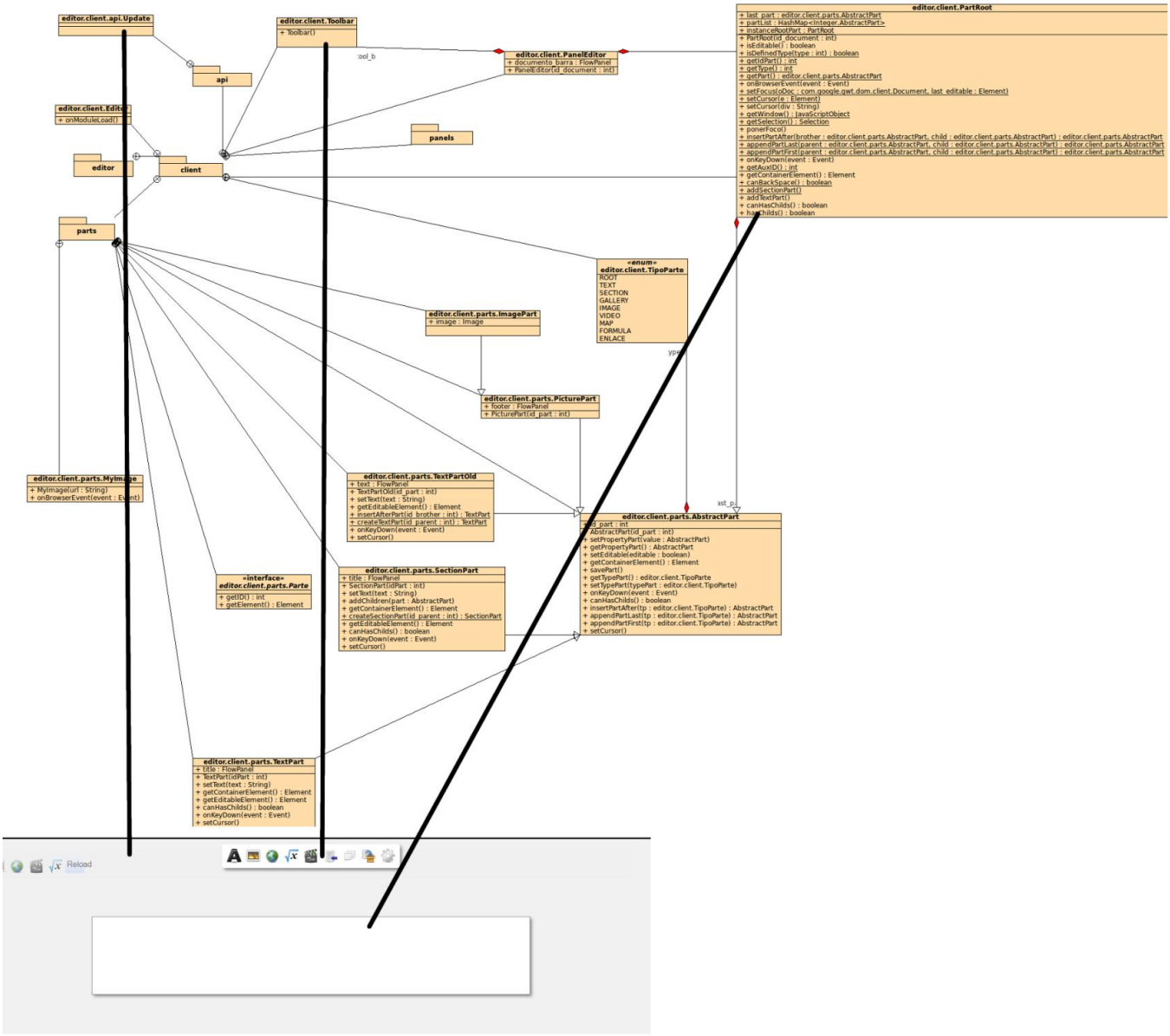

Ilustración 6-27 Módulo editor de documentos

El editor de documentos está formado por tres paneles principales que contienen el resto de elementos:

- Panel de estructura del documento
- Panel superior con los tipos de parte
- Área de edición.

6.5 Servidor

El servidor tiene mayor complejidad que el cliente. A alto nivel, podemos diferenciar dos partes principales en el lado del servidor:

- El servidor *Web* utilizado es *Apache2*. Consta de un módulo *PHP* 5.3. y un servidor de bases de datos *MySQL*. Estos elementos han sido meramente instalados y configurados.
- En la capa superior tenemos la *API.* Este módulo contiene funciones de PHP de alto nivel que sirven para manejar documentos y partes. Esta capa es particular de TreeDoc.

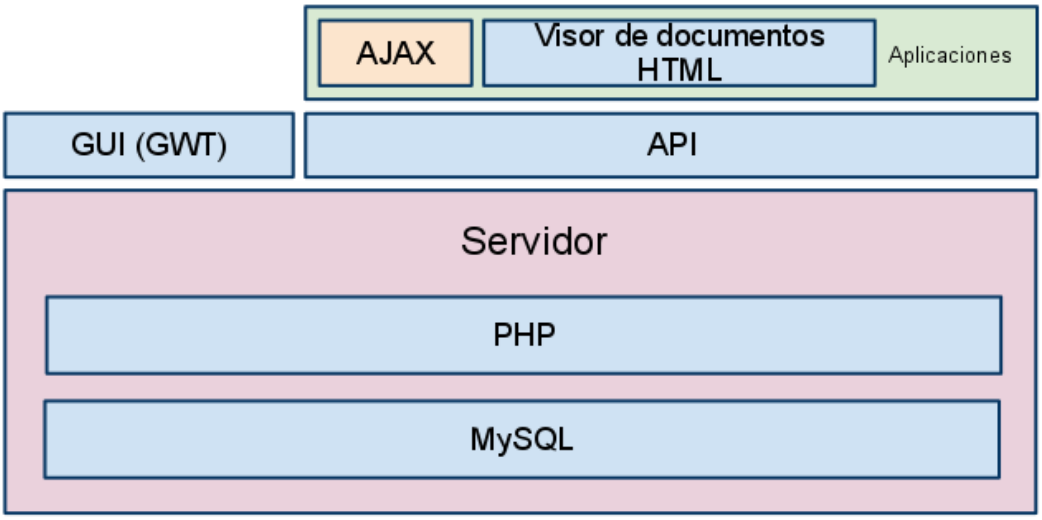

Ilustración 6-28 Esquema del servidor

6.6 Comunicación cliente-servidor

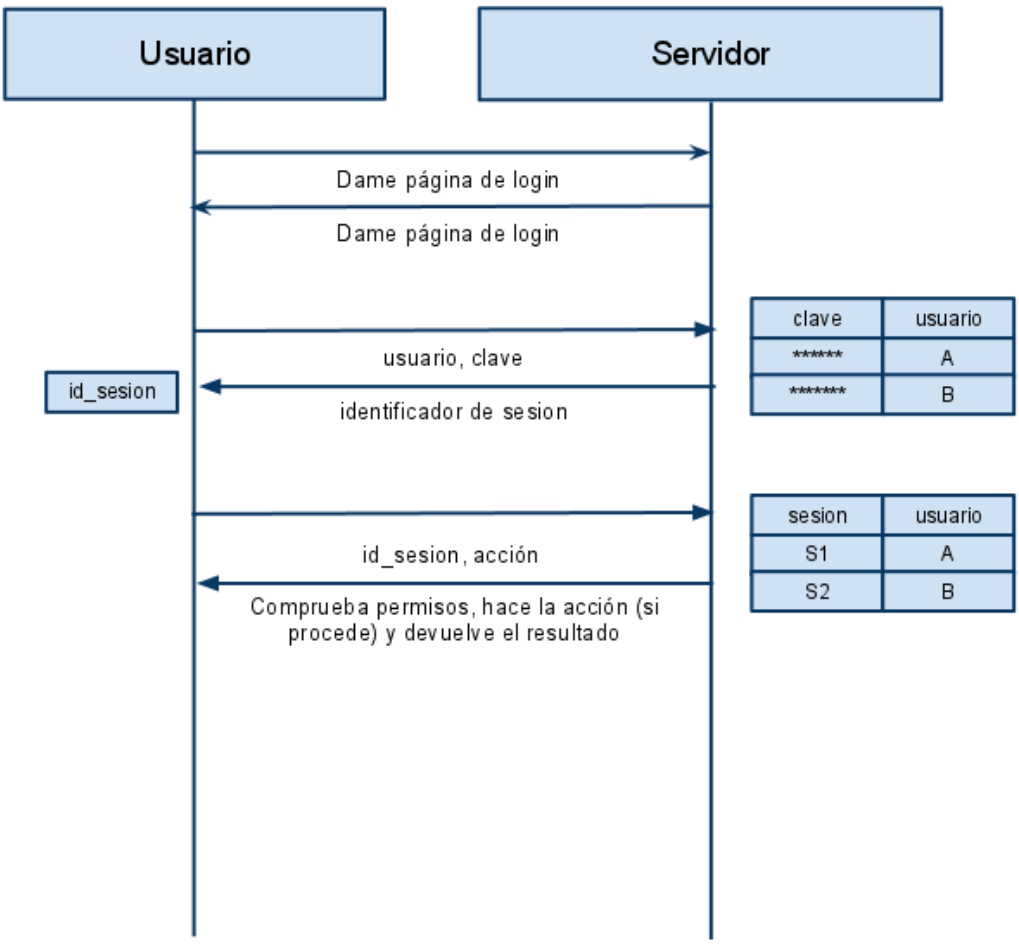

Ilustración 6-29 Esquema de comunicación cliente - servidor

El diagrama anterior corresponde al proceso de autenticación de un usuario. Se trata de un ejemplo de la comunicación entre cliente y servidor usado para entender fácilmente cómo se lleva a cabo esta comunicación:

1a.- El usuario hace una petición al servidor pidiendo la página de *login*.

1b.- El servidor contesta al usuario enviándole la página de *login*.

2a.- El usuario envía su nombre de usuario y su clave al servidor

2b.- El servidor devuelve un identificador de sesión

A partir de este momento, el usuario siempre envía un identificador de sesión y una acción.

El servidor comprueba en cada petición del usuario si dicho usuario tiene los permisos necesarios para realizar la acción indicada. Después, devuelve el resultado correspondiente al usuario.

El sistema permite abrir y cerrar sesiones de trabajo y obtener información básica del usuario al que pertenece la sesión en curso. Cuando un usuario intenta llevar a cabo una acción, se comprueba si tiene permisos suficientes para ella, obteniendo información de la sesión que ha abierto y comprobando si tiene los permisos necesarios.

Más adelante se presentan diagramas de secuencia detallados.

6.7 Modelo de datos

Después de haber descrito cómo funciona el sistema a alto nivel, se describe el modelo de datos utilizado. Esto comprende la capa de persistencia de los datos, y la capa de acceso a datos.

6.7.1 Capa de datos: Base de datos MySQL

La tecnología utilizada para la persistencia en TreeDoc es MySQL 5.1.

Para tomar esta decisión se tuvo en cuenta:

- Licencia *GNU GPL* (excepto para empresas que lo incorporan a productos privativos).
- Su uso está muy extendido en aplicaciones *Web.*
- Gratuidad.
- Amplia comunidad y soporte.
- Facilidad de uso
- Uso casi transparente por parte de PHP.

En el Apéndice B: Tecnologías y herramientas utilizadas” se amplían algunas características importantes de *MySQL*.

Las ilustraciones de esta sección muestran diagramas con las tablas de la base de datos y las siguientes relaciones existentes entre ellas:

- Un grupo tiene puede tener 0 o varios usuarios.
- Un usuario puede no pertenecer a ningún grupo o pertenecer a varios grupos a la vez. Por defecto, cuando se crea un usuario se añade al grupo “Todos”.
- Hay dos usuarios especiales:
    - guest: usuario con id 0; es el usuario invitado, con el que se accede por defecto cuando no se ha iniciado una sesión. Este usuario sólo puede acceder a la pantalla de login.
    - sadmin: Usuario especial con todos los privilegios.
- Un documento puede tener una o más partes. Al crear el documento, se crea por defecto una parte llamada ROOT. A partir de esta parte, se pueden introducir las demás.
- Una parte pertenece a un único documento.
- Una parte puede tener 0 o más comentarios.

Para facilitar la comprensión del diseño de la base de datos, su descripción se ha separado en dos diagramas, pero representan partes de una única base de datos.

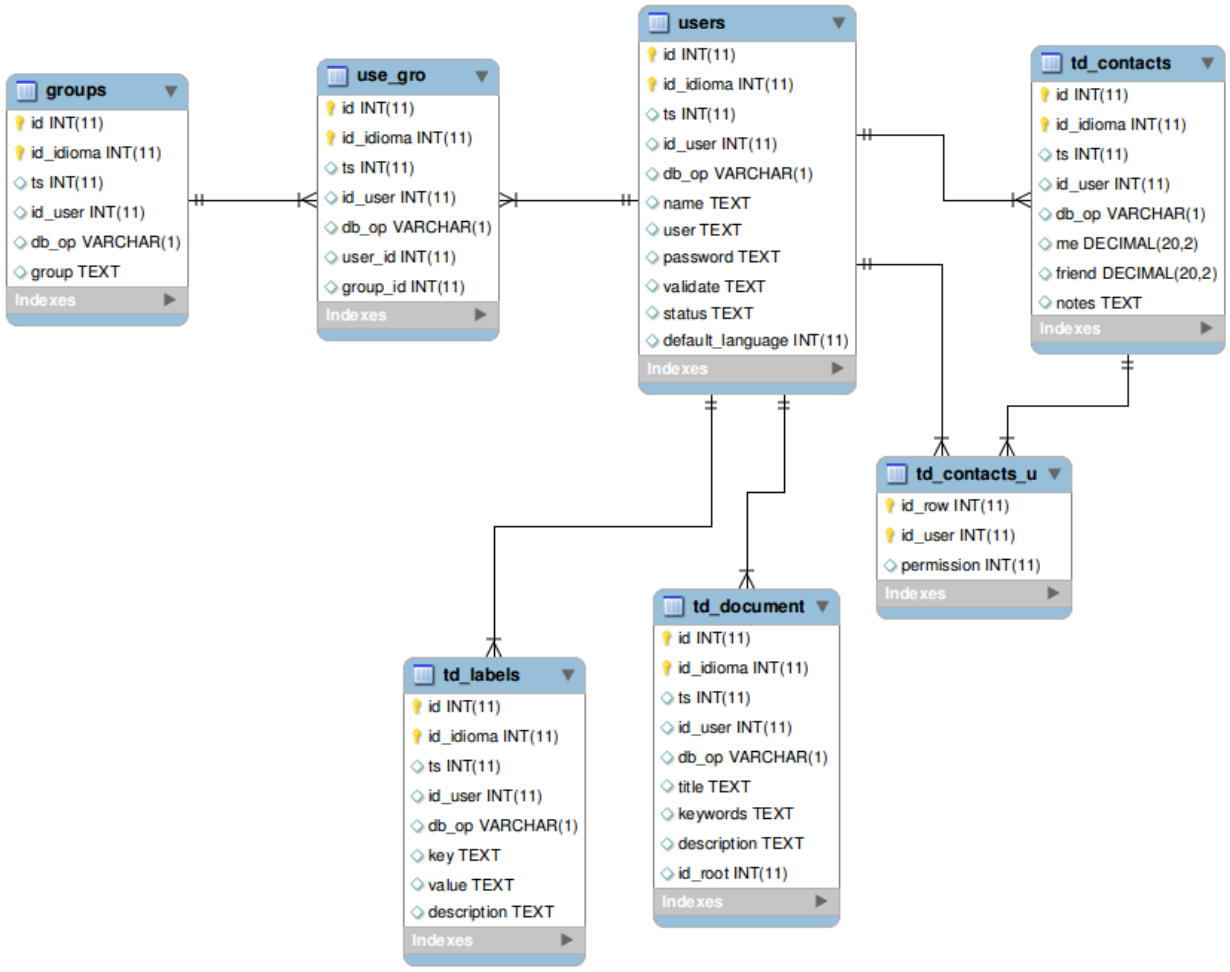

Ilustración 6-30 Diagrama de base de datos de contactos y usuarios

El diagrama anterior muestra las *tablas* de *usuarios*, *grupos* y *contactos*, y algunas tablas relacionadas con éstas. La tabla use*_gro* se ha creado para almacenar la relación entre usuarios y grupos, ya que es una relación n-m.

La tabla que almacena los usuarios es *Users*, y tiene campos para almacenar toda la información del usuario, además de un identificador de usuario que se usa como clave externa de otras tablas para relacionar al usuario.

La tabla *Contacts* almacena la información de los contactos, y se ha creado la tabla td*_contacts*_u para representar la relación entre usuarios y contactos, ya que de nuevo es una relación n-m.

La tabla *Labels* sirve para almacenar las etiquetas de los menús y opciones de la aplicación en distintos idiomas, y según el idioma del usuario, mostrar unas u otras.

Por último aparece la relación entre *Users* y td_*document*. La tabla td_*document* se explica con mayor detalle sobre la Ilustración 6-31, en este punto únicamente se indica que contiene un campo que es una clave externa que hace referencia al identificador de usuario que crea el documento.

El diagrama de la Ilustración 6-31 muestra las tablas de usuarios (*users*), documentos (td_*document*) y partes (*td*_part), y otras tablas relacionadas con éstas.

La tabla td_*document* está relacionada con las tablas td_*document*_h, td_*document*_g y td_*document*_u, que almacenan el historial de documentos, los permisos de cada grupo sobre cada documento y los permisos de cada usuario sobre cada documento.

La tabla *td_part* también está relacionada con la tabla de documentos, y contiene un campo para almacenar el identificador del documento al que pertenece (el documento en el que se crea la parte por primera vez, aunque luego se enlace desde otros documentos). También se relaciona con la tabla *users* ya que debe almacenar el usuario que la crea, y tiene una tabla de historial asociada (*td_part_h*).

El campo *data* almacena el contenido de cada parte en forma de datos serializados, es decir, codificados de tal forma que se pueden almacenar varios campos de información.

Se ha añadido el campo *image* en la misma tabla para que lo utilicen algunos tipos de parte (imagen, mapa, vídeo, fórmula). Este campo está relacionado con una tabla nueva de imágenes.

La tabla *images* e *images_cache* ofrecen una forma de almacenar imágenes y de pre-procesarlas para cambiar su tamaño. Básicamente la idea es asociar un identificador a cada imagen, guardarlas en un archivo en disco (no en la base de datos). Además, cada imagen se puede pedir con tamaños diferentes, así en cada nueva petición se procesan y se guardan en la tabla *images_cache*.

Los campos *id_row* e *id_group* que aparecen en los diagramas son generados por la tabla abstracta para añadir permisos (de usuario y de grupo) a nivel de registro.

Ver Apéndice B: Modelo de permisos extendido con grupos para obtener información detallada sobre la generación de los permisos.

Para representar los permisos se utilizan números enteros de la siguiente manera:

Los permisos se codifican mediante bits en un entero. Para los registros se utilizan 4 permisos (lectura, eliminación, modificación y administración) así que necesitamos 4 bits. Cada permiso se codifica con un número entero que corresponde a una configuración binaria en la que sólo un bit está a 1:

```
const PERMISSION_REGISTER_READ = 1;
const PERMISSION_REGISTER_UPDATE = 2;
const PERMISSION_REGISTER_DELETE = 4;
const PERMISSION_REGISTER_ADMIN = 8;
```

En binario son, respectivamente:

00000001
00000010
00000100
00001000

De esta forma si un registro debe tener permiso de lectura y de eliminación se hace la operación binaria OR sobre las constantes de los permisos:

```
00000001 OR_binario 00000100 = 00000101
1 OR_binario 4 = 5 (en decimal)
```

Y se almacena el entero correspondiente.

En el caso de una lectura, si se tiene el permiso, se haría:

```
00000101 AND_binario 00000001 = 00000001
```

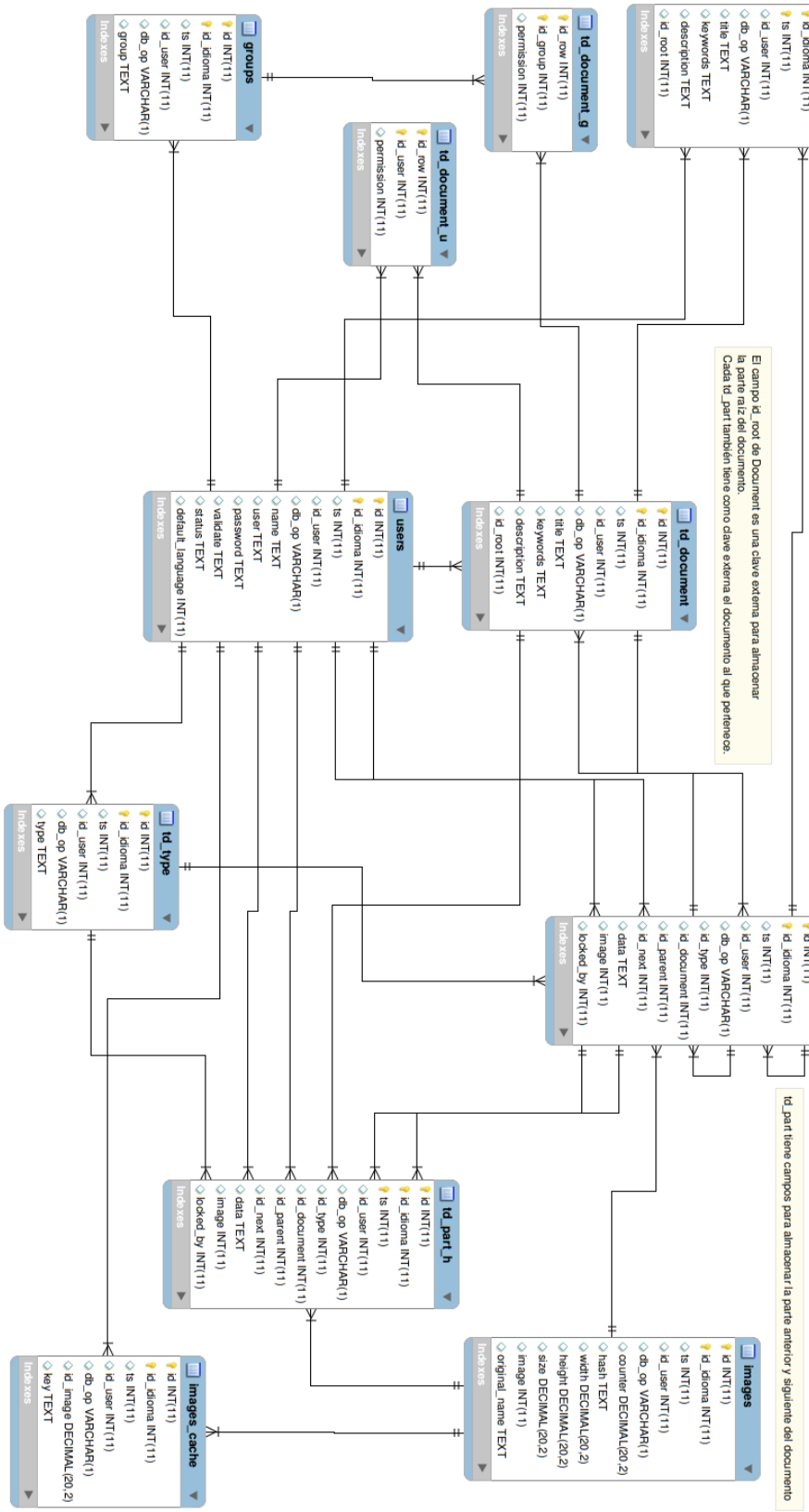

Ilustración 6-31 Diagrama de base de datos de documentos

6.7.2 Capa de acceso a datos

La capa de acceso a datos proporciona abstracción sobre la base de datos utilizada, haciendo que la lógica de la aplicación sea independiente de la base de datos concreta que se utiliza. Consiste en una tabla abstracta creada en PHP, a través de la cual la capa lógica accede a los datos. De esta forma, si se producen cambios en la base de datos, no afectarán a la lógica.

Ya existen soluciones de este estilo (por ejemplo *Propel* o *Doctrine*) pero ninguna integra de forma simultánea permisos a nivel de registro, múltiples idiomas, e historial. Estas soluciones son tan complejas que modificarlas para adaptarlas a nuestro sistema supone un coste muy alto. Algunas soluciones incluso necesitan ser "compiladas" o traducidas a PHP cada vez que se cambia el modelo.

En nuestro sistema, creamos una capa de acceso a los datos mediante la clase *Database*, y las clases *Table, TableDefinition, TableQuery* y *Register*.

6.7.2.1 Database

La clase *Database* abstrae la conexión a la base de datos. TreeDoc sólo trabaja con un sistema gestor de base de datos (*SGBDR*) al mismo tiempo, por lo tanto sólo tiene sentido una única instancia del objeto *Database*. Para asegurar la existencia de una sola instancia, se ha decidido la implementación de esta parte del sistema mediante el patrón de diseño *Singleton*(11). Este patrón proporciona una solución para asegurar que se instancia una sola vez una clase concreta, y el resto de clases acceden a esta instancia.

Para seguir el diseño propuesto por el patrón *Singleton*, el constructor y el método *__clone()* son privados. De esta forma, no se permite que una clase externa cree o clone un objeto de la clase *Database*. Se implementa la función *getInstance().* Cuando una clase realiza una llamada a esta función, se comprueba si existe ya una instancia, en cuyo caso devuelve la instancia existente. Si no es así, crea una instancia y la devuelve.

La clase *TreeDoc* es la encargada de llevar a cabo la inicialización de la instancia de *Database,* que proporciona la función *getInstance()* (explicada anteriormente) para que el resto de clases de los distintos módulos puedan acceder a esta instancia.

6.7.2.2 Clases Table, TableDefinition, TableQuery y Register

Hay cuatro clases que junto con *Database* forman la capa de acceso a datos. Estas clases interactúan con el resto de módulos del sistema abstrayendo la base de datos utilizada por debajo. Estas clases son *Table*, *TableDefinition*, *TableQuery* y *Register*.

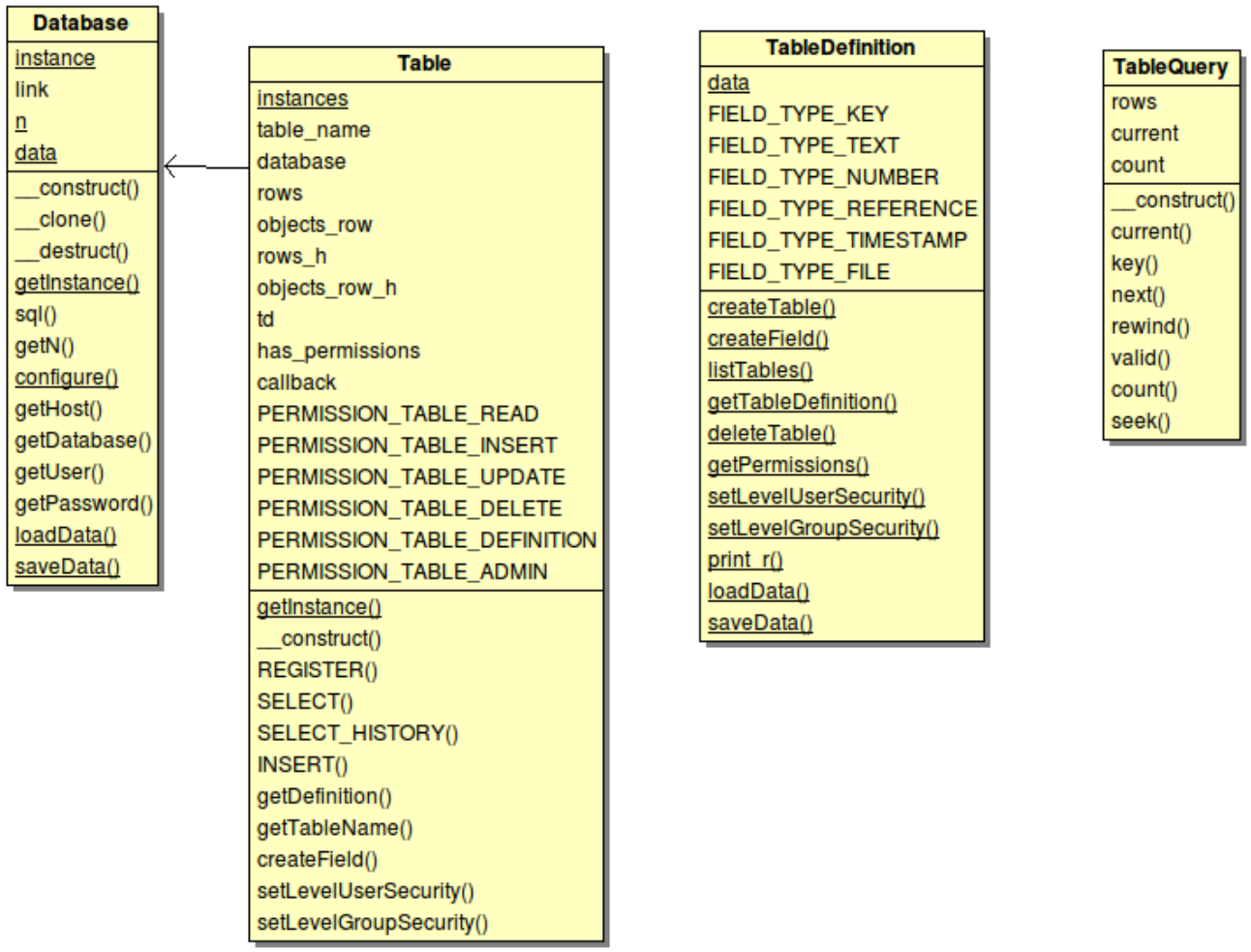

Ilustración 6-32 Diagrama de clases Table, TableDefinition, TableQuery y Register

La clase *Table* implementa el patrón *Multiton* (11). Esto significa que existen varias instancias de la clase *Table*, pero cada instancia tiene un nombre diferente. De este modo tenemos un objeto de este tipo para cada tabla de la base de datos:

`Table::getInstance('td_users')`, `Table::getInstance('td_documents')` *etc.* Cada instancia de *Table* está asociada a una instancia de la clase *TableDefinition*.

TableQuery se utiliza para devolver los resultados de una consulta de selección sobre una tabla de la base de datos. Cuando se realiza una consulta sobre una tabla, se almacenan los datos y se encapsulan en objetos *Register*, se crea una instancia de la clase *TableQuery*, que contiene las referencias a los objetos *Register* correspondientes tras ejecutar la consulta. Si se realizan varias consultas sobre la misma tabla no se crean objetos *Register* duplicados.

El siguiente ejemplo ilustra el funcionamiento de las clases Table, TableDefinition, TableQuery y Register. En el ejemplo se realizan cuatro consultas:

La primera tiene como resultado R1, R2 y R3 de la tabla 'usuarios'.

La segunda R1 y R2 también de 'usuarios'. Como ya están almacenados en objetos *Register* por la consulta anterior, no se crean nuevos.

La tercera R1 y R2 de 'coches'.

La cuarta R1, R2 y R3 de 'coches'. Como R1 y R2 ya tienen objetos *Register* de la consulta anterior, sólo se crea un objeto Register para R3.

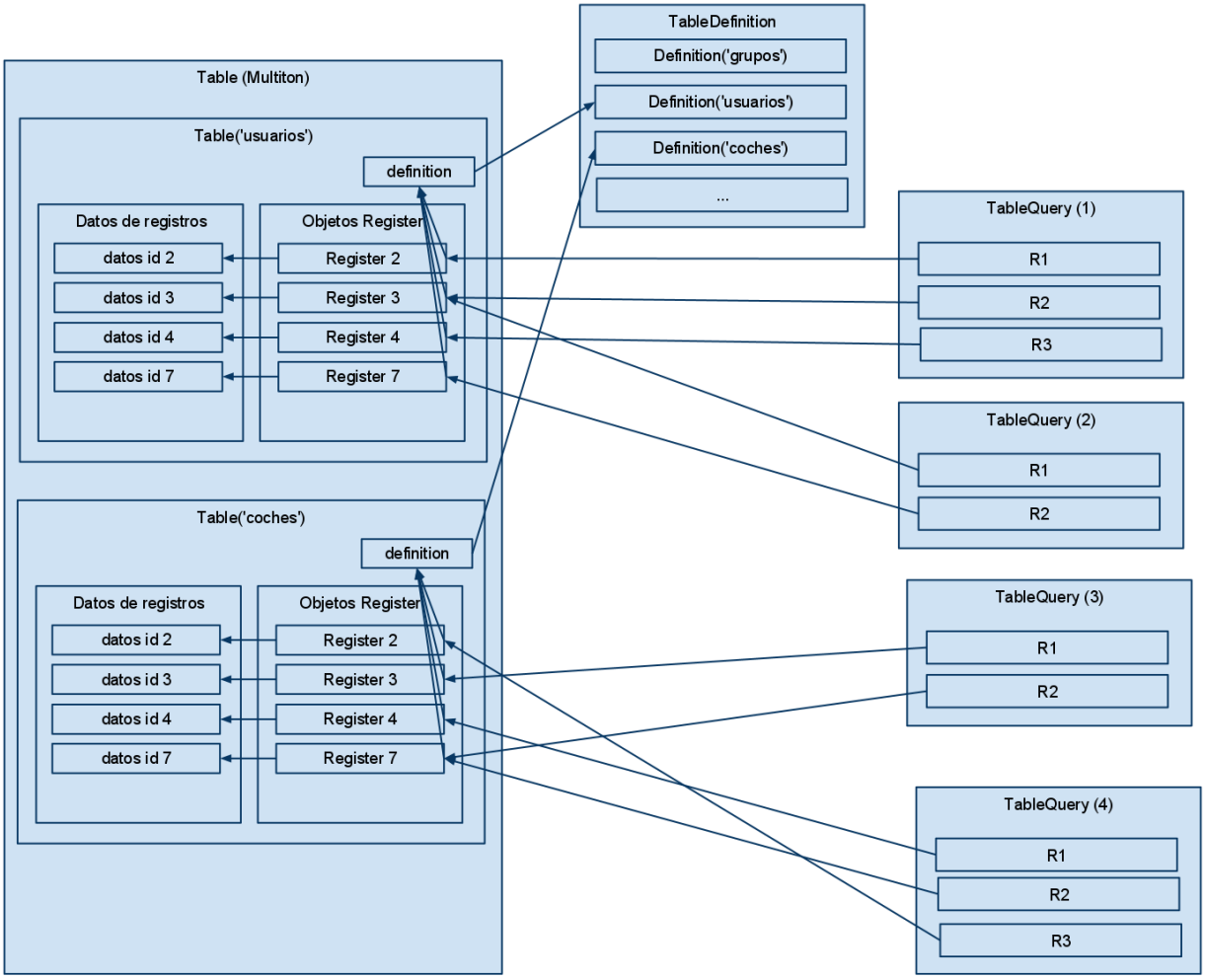

Ilustración 6-33 Ejemplo funcionamiento de las clases Table, TableDefinition, TableQuery y Register

La Ilustración 6-34 relaciona *Document*, *Table*, *TableDefinition* y *FileStorable*:

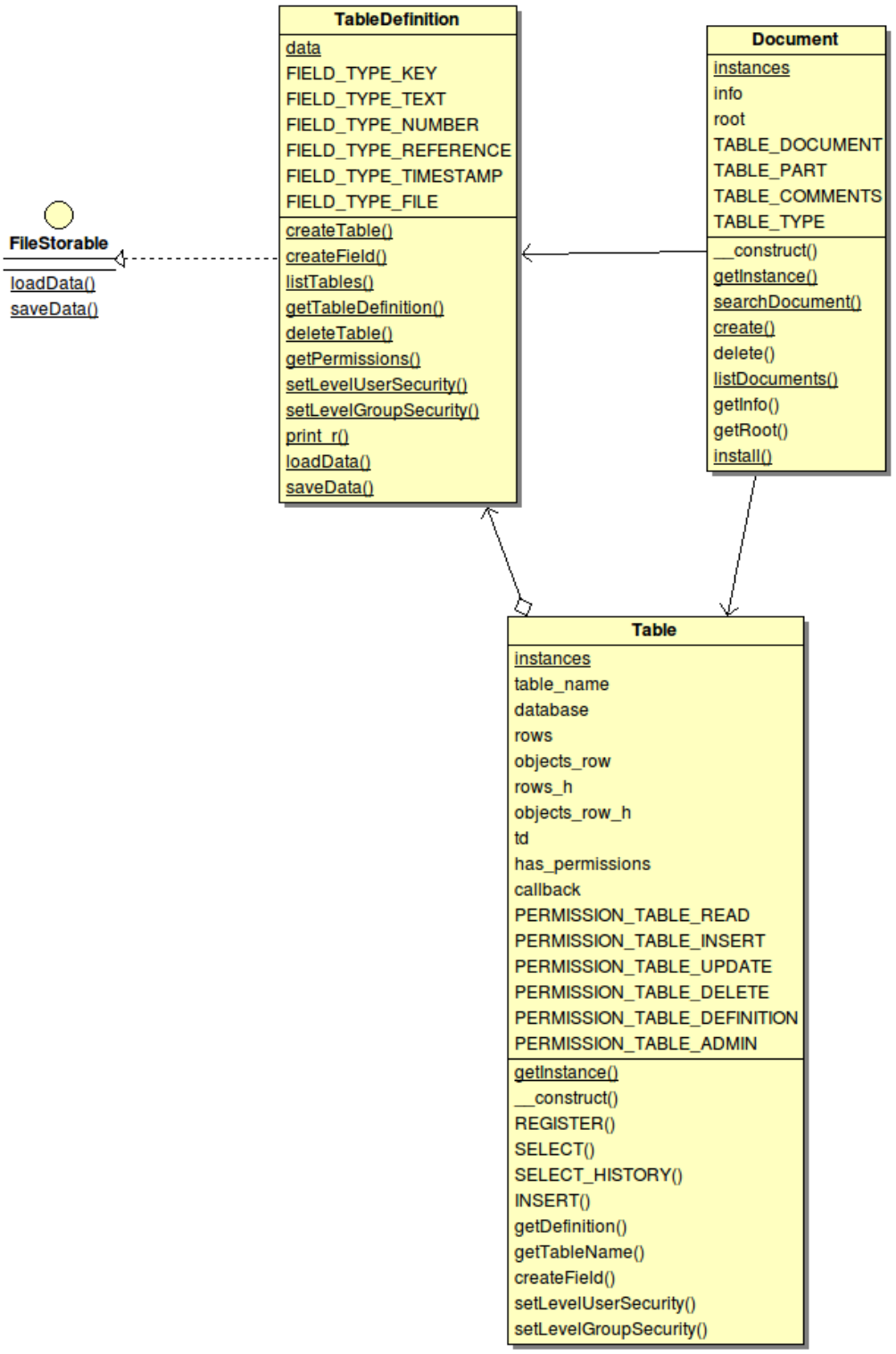

Ilustración 6-34 Diagrama de clases de Document, Table, TableDefinition y FileStorable

En la Ilustración 6-34 se pueden ver la clase que gestiona la estructura de las clases: *TableDefinition*. *Table* utiliza la definición para manipular los datos. La clase *Document,* que también aparece en la ilustración citada, utiliza instancias de las dos anteriores para almacenar sus datos.

6.7.3 Capa de negocio

La capa de negocio tiene la lógica de la aplicación. Está implementada en PHP y la forman los módulos *Documents*, *Users*, *Groups*, y *Sessions*.

6.7.3.1 Documents

El módulo de Documentos se encarga de la gestión de documentos y partes. Incluye las clases:

- Document
- DocumentPart
  - Clase abstracta que representa una parte de un documento.
- DocumentPartRoot
  - Parte “raíz” de un documento.
- DocumentPartDimensionable
  - Clase abstracta para representar las partes que están contenidas en un cuadrado y cuyas dimensiones pueden ser modificadas.
- DocumentPartFormula
  - Parte de tipo fórmula.
- DocumentPartImage
  - Parte de tipo imagen.
- DocumentPartInclude
  - Parte que es un enlace a una parte de otro documento.
- DocumentPartSection
  - Representa una sección dentro de un documento (un título). Puede tener partes anidadas.
- DocumentPartVideo
  - Representa una parte de tipo video.
- DocumentPartText
  - Parte de tipo texto.
- DocumentPartTDF
  - Clase estática que permite añadir información en formato TDF a una parte.
  - Esta clase es útil para realizar copias de seguridad de los documentos.
- DocumentPartGWT
  - Clase estática que permite añadir información que viene de la interfaz gráfica GWT a una parte.

- Permite abstraer el tipo de parte. De esta forma se consigue que sólo sea necesario un método Ajax para leer y escribir.

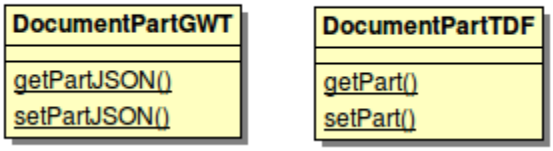

Ilustración 6-35 Diagrama de clases DocumentPartGWT y DocumentPartTDF

La clase principal de este módulo es Documentos.

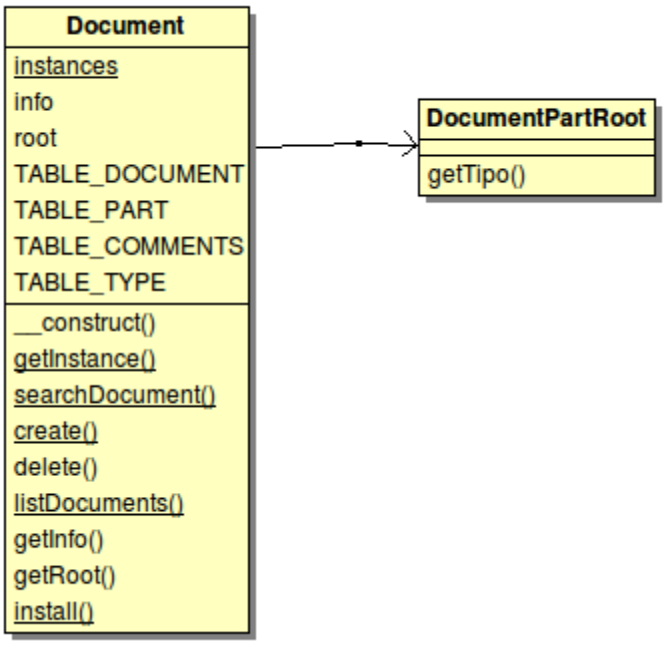

Ilustración 6-36 Diagrama de clases Document y DocumentPartRoot

Del mismo modo que ocurre la clase *Table,* explicada anteriormente, su implementación sigue el patrón de diseño *Multiton* para controlar la creación de instancias de esta clase. Para ello, se mantiene un diccionario que mapea claves con objetos únicos.

Para aplicar este patrón en nuestra clase *Documents*, tenemos un atributo dentro de la clase que es un *array* de instancias. El constructor es privado, y tenemos una función *getInstance(id)*,

que al instanciar la clase verifica si la clave existe en el *array* de instancias. Si ya existe, devuelve una referencia a la instancia del *array* con el identificador dado, y si no, crea una nueva instancia con dicho identificador.

Para manejar internamente lo documentos, éstos se guardan con una estructura en forma de árbol. Al crear el documento, éste se crea con una parte *Root* o raíz, de tipo *DocumentPartRoot*, y el resto de partes se van añadiendo a esta raíz. La tabla *td_part* tiene un atributo para almacenar el documento al que pertenece. Para ver más en detalle cómo se relacionan las partes y los documentos se puede ver la Ilustración 6-38 que se explica más adelante.

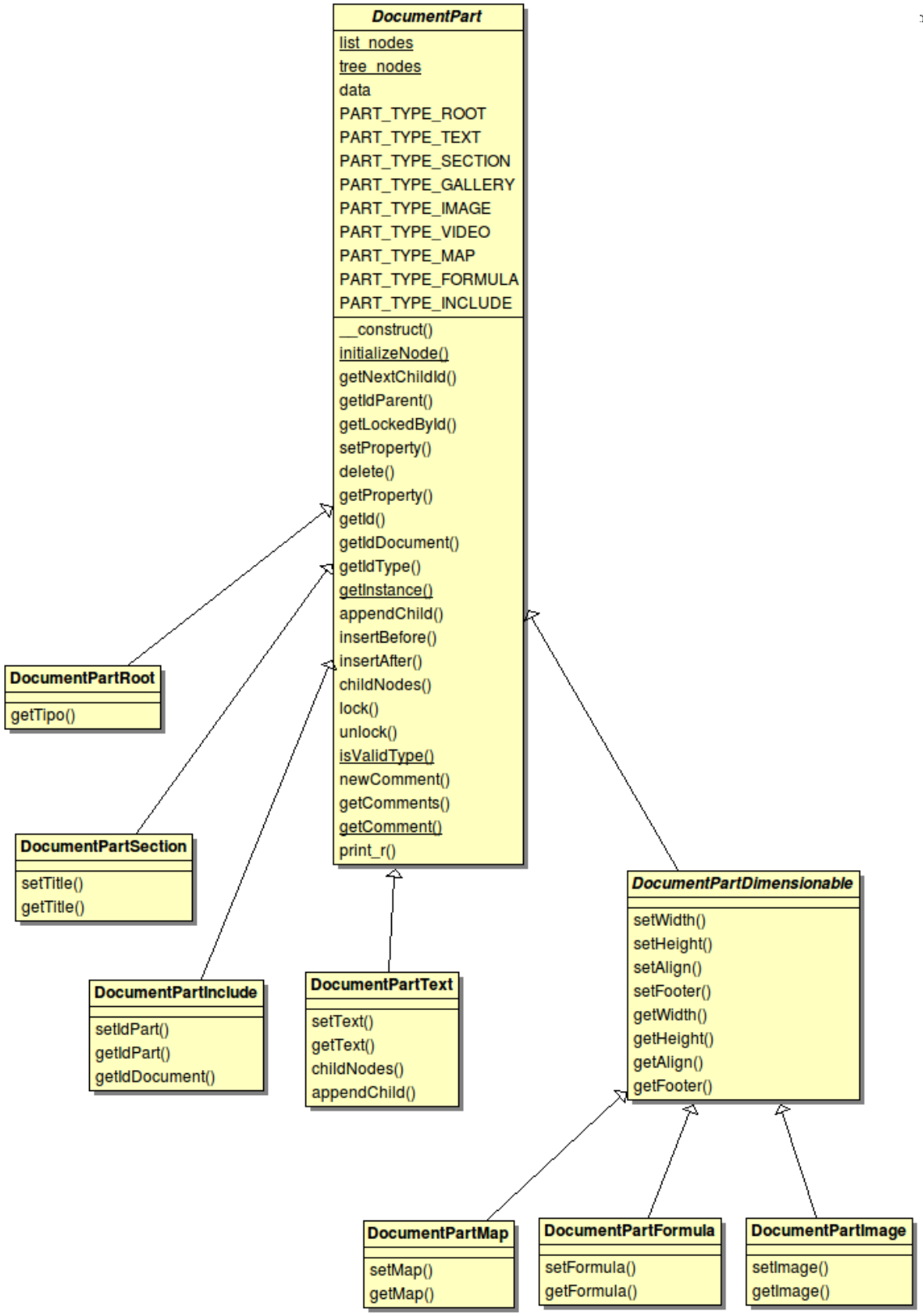

Ilustración 6-37 Diagrama de clases de Documentos y partes de documentos

La Ilustración 6-37 muestra la relación entre las clases de *DocumentPart* y las que heredan de ésta. Más adelante se explican en detalle las características de cada parte. De momento, se tratan a alto nivel para mostrar el funcionamiento general de esta parte de la aplicación.

Cuando se añade una nueva parte al documento, internamente se añade un nodo al árbol. Hay tipos de partes que pueden tener a su vez partes anidadas, y otros tipos que no. Esto se controla gracias a un método para añadir nodos al árbol. Este método se define en la clase *DocumentPart:*

`appendChild`()

En la Ilustración 6-37 se puede ver que algunas de las clases que heredan de *DocumentPart* sobrescriben este método. Las clases que lo sobrescriben tienen la implementación vacía. De este modo se protege el uso indebido del método en el caso de las clases que no pueden tener partes anidadas.

La estructura arbórea de partes se crea de tal forma que cada nodo hijo apunta al padre. Además, también se crean relaciones entre los nodos de cada nivel formando una lista enlazada. De esta manera se lleva a cabo el control del orden de las partes. La Ilustración 6-38 muestra estas relaciones entre nodos. En dicha imagen se puede observar también, que cada parte incluye una referencia al documento que la contiene. Esto se ha hecho así por motivos de eficiencia.

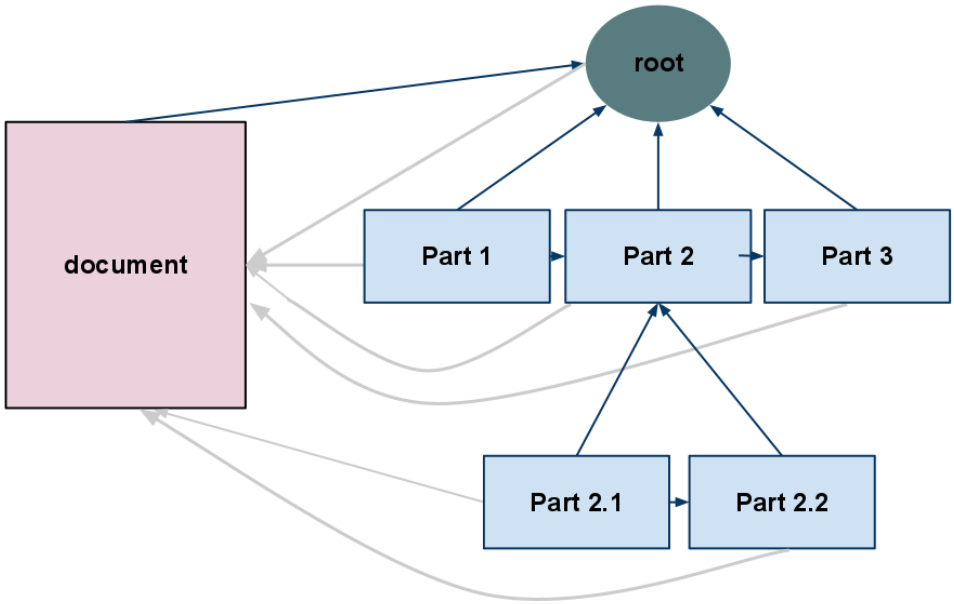

Ilustración 6-38 Estructura en árbol de un documento y sus partes

6.7.3.2 Usuarios

La gestión de usuarios se lleva a cabo mediante la clase *Users* principalmente. Esta clase es una aplicación del patrón de diseño *Adapter* o *Wrapper*(11). Este patrón permite abstraer el funcionamiento de las tablas de la base de datos y crear una capa por encima que sirva para la gestión de los usuarios. Proporciona funciones para crear, listar y validar usuarios.

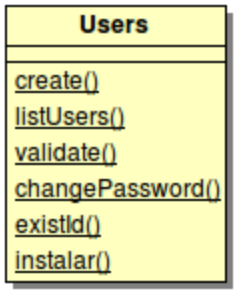

Ilustración 6-39 Clase Users

El acceso a la totalidad de los campos de la tabla *users* se hace mediante la tabla abstracta de la forma `Table::getInstance('users')`. Aunque también se puede acceder mediante la tabla abstracta, la clase *Users* es necesaria para el funcionamiento de la propia tabla abstracta.

6.7.3.3 Contactos

Un usuario puede añadir o borrar contactos; esto es, puede elegir a otros usuarios como "contactos" y guardar una referencia de ellos. Para implementar esta funcionalidad, tenemos la clase *Contact,* que permite llevar a cabo las funciones de gestión de contactos como añadir o eliminar, y además, permite añadir información relacionada con el contacto, como *Estado del contacto* y *notas del contacto.*

Ilustración 6-40 Clase Contacts

6.7.3.4 Grupos

La gestión de los grupos de usuarios tiene como clase principal *Groups.* Su implementación, al igual que *Users,* es una aplicación del patrón *Adapter o Wrapper.*

La clase *Groups* proporciona las funciones para crear grupos, añadir/borrar usuarios a/de un grupo, listar los usuarios de un grupo o comprobar si existe un grupo con un identificador dado.

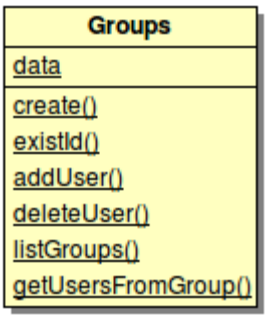

Ilustración 6-41 Clase Groups

6.7.3.5 Sesiones

La clase *Session* contiene la lógica necesaria para gestionar las sesiones. Las sesiones son una forma de almacenar información sobre los usuarios que están en un momento dado utilizando la aplicación.

En la sesión de un usuario se almacenan datos como su nombre de usuario y su idioma. Estos datos se almacenan en unas variables denominadas variables de sesión.

Cada vez que se accede al sistema se genera un identificador de sesión único, que sirve para saber las variables de sesión que pertenecen a la sesión de trabajo en curso.

Es importante puntualizar respecto a las sesiones que éstas se almacenan directamente en el disco del servidor, sin pasar por la base de datos.

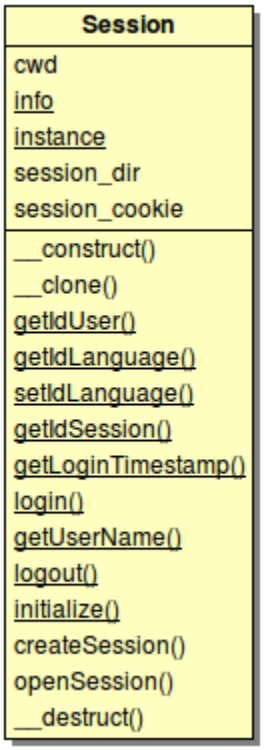

Ilustración 6-42 Clase Session

La Ilustración 6-43 muestra las relaciones de las clases que implementan el interfaz *FileStorable*.

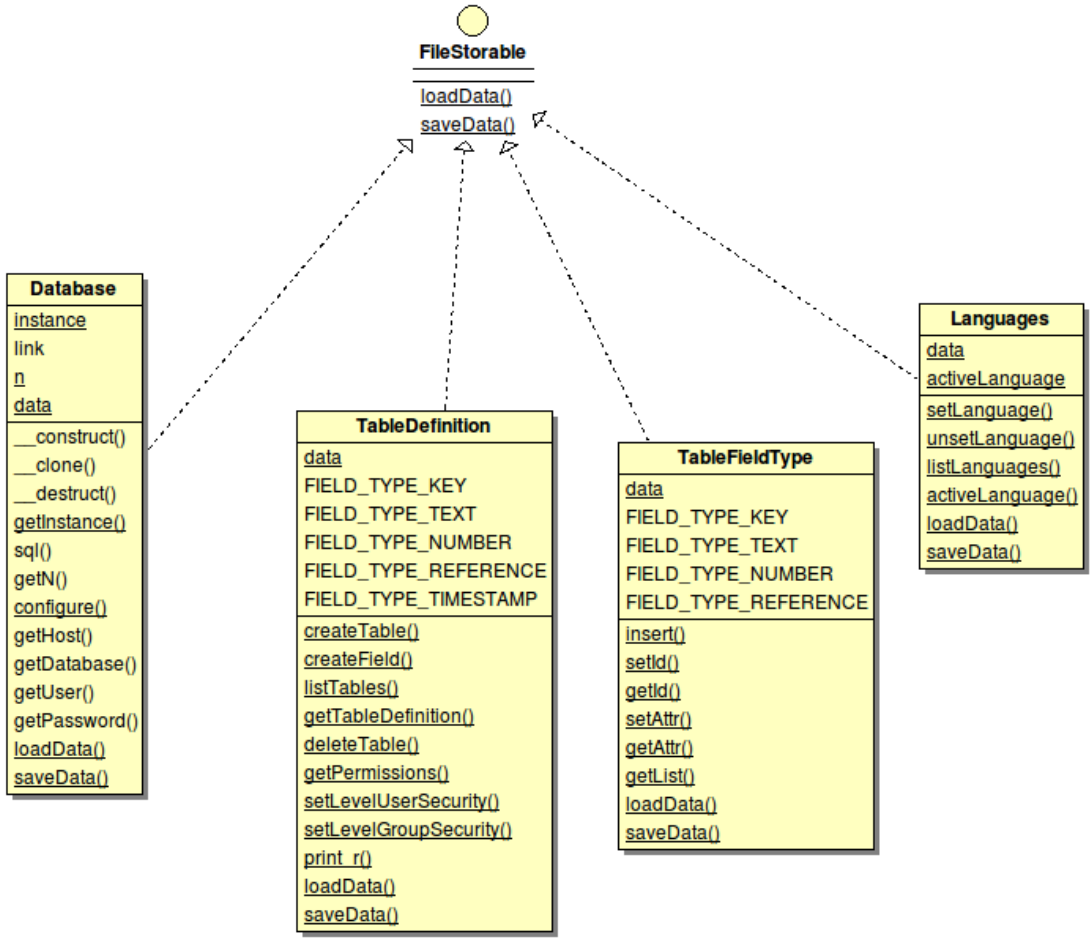

Ilustración 6-43 Diagrama de clases que implementan Filestorable

6.8 Comportamiento dinámico del sistema

Las relaciones entre objetos se explican mediante diagramas de secuencia. Se han modelado mediante estos diagramas las acciones más representativas que se pueden llevar a cabo.

Está centrado en la parte del servidor, por lo que el resto del sistema se ha simplificado y se representa mediante *GUI* (*Graphic User Interface*) y *API*. De esta manera se facilita la comprensión de los diagramas.

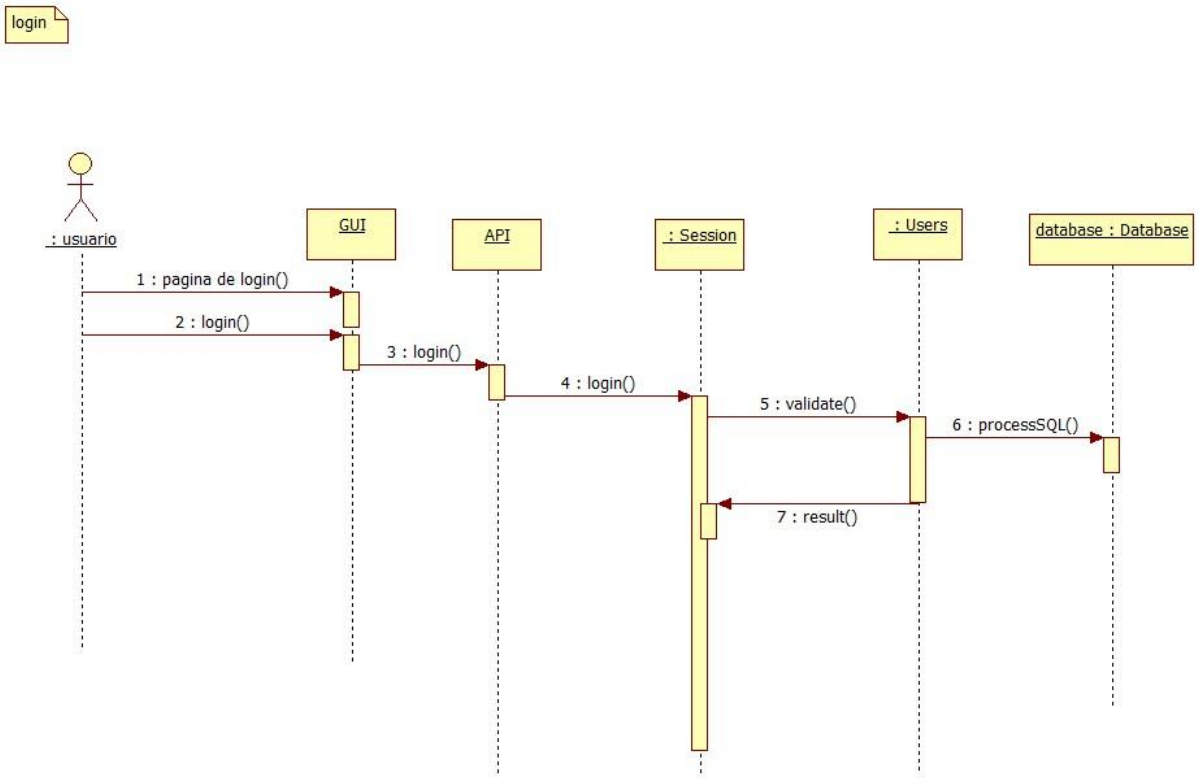

Ilustración 6-44 Diagrama de secuencia de Login

La Ilustración 6-44 muestra el Diagrama de secuencia de *login (iniciar sesión)*. Se trata de un diagrama a alto nivel para mostrar las clases implicadas. Los parámetros necesarios que un usuario debe introducir en este caso son el nombre de usuario y la contraseña. Estos datos se introducirán en la interfaz gráfica (*GUI) y* se pasarán al servidor a través de la API. Una vez en el servidor, la clase *Session* es la encargada de pasar estos datos a *Users* para que sean validados. *Users* realizará una consulta a *Database* y ésta será quien determine si los datos corresponden a un usuario válido o no.

Si los datos son válidos, se entrará a la aplicación (concretamente a la parte de gestión de documentos). Si no son válidos, se informará al usuario mediante un mensaje.

6.8.1.1.1 Buscar un documento

El buscador de documentos permite buscar un documento por título. Se puede escribir el título del documento o parte de él. Se realizará una llamada a la función *searchDocument($search_string)* de la clase *Document,* y en el parámetro *$search_string* se

debe enviar la cadena de búsqueda(título o parte de él). El diagrama de la Ilustración 5-1 muestra la interacción de los objetos a partir de esa llamada a la clase *Document:*

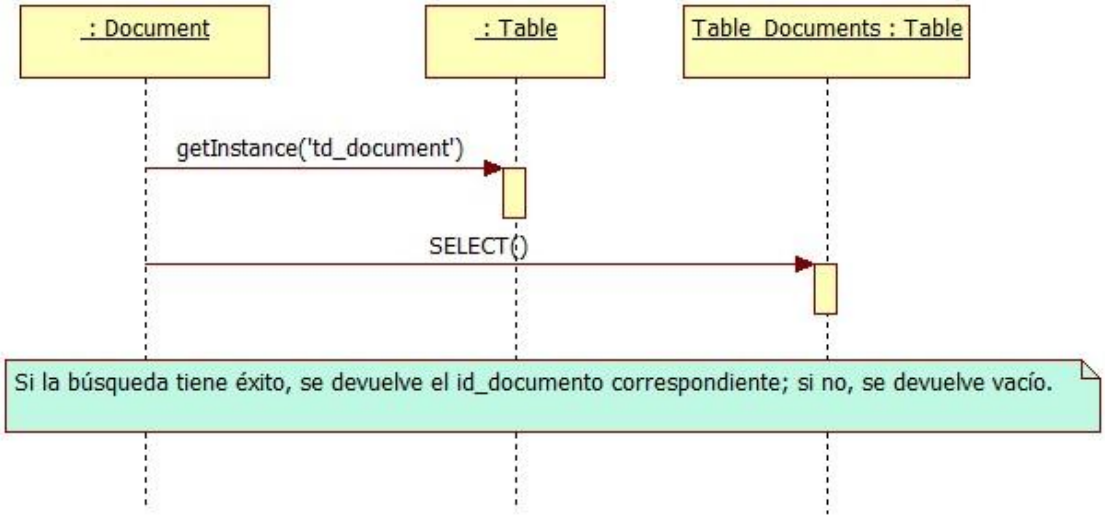

Ilustración 6-45 Diagrama de secuencia de búsqueda de documento

En el diagrama se observa como la clase *Document* realiza una llamada a *Table* para obtener la instancia de la tabla de documentos. La clase *Document* construye la cadena SQL (*Structured Query Language*) que después enviará en la función *SELECT* a la instancia de la tabla de documentos. Esta función devolverá el identificador del documento (*id_document)* que cumple los requisitos de búsqueda, y si no existe ninguno, se devolverá vacío.

6.8.1.1.2 Crear un documento nuevo

Cuando se selecciona la opción de crear documento en el editor, esta acción se procesa y la API generará una llamada a la función *create()* de la clase *Document.*

En el diagrama de la Ilustración 6-46 se describe la interacción de las clases implicadas; Se ha simplificado únicamente nombrando la función *INSERT,* que se detalla en el diagrama de la Ilustración 6-47.

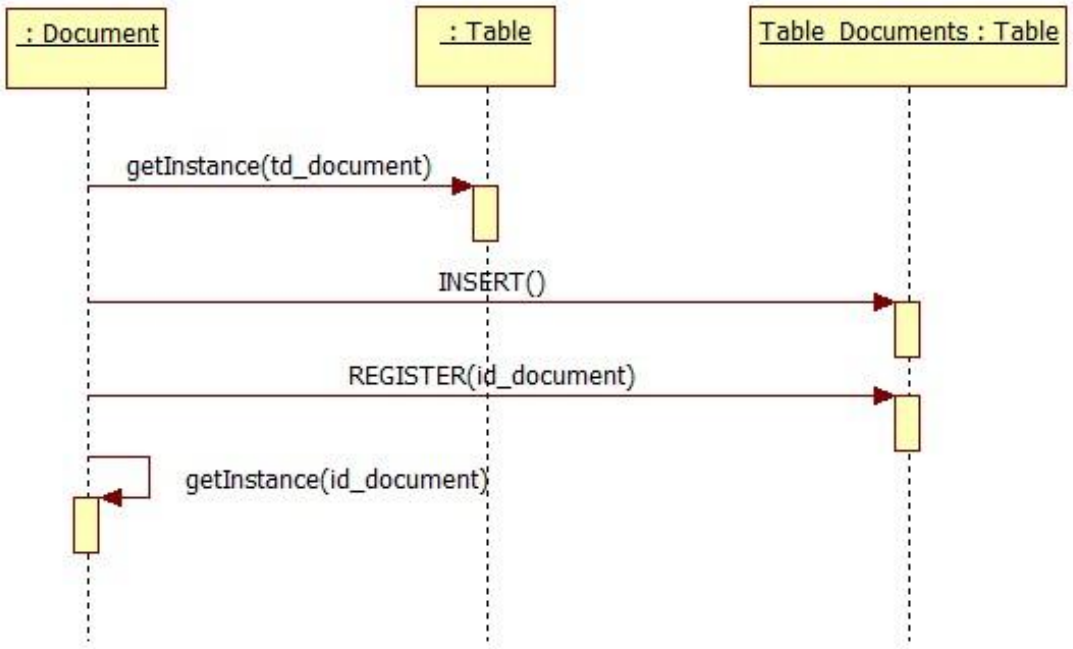

Ilustración 6-46 Diagrama de secuencia de crear documento

La clase *Document* utiliza la función *getInstance()* de la clase *Table* para obtener la instancia de la tabla de documentos. Sobre esta tabla, se realiza la llamada a *INSERT (*verIlustración 6-47). Con la llamada posterior a *REGISTER* se obtiene la instancia del documento creado, y se inicializan los valores de este documento.

Finalmente se realiza una llamada a *getInstance($id_doc)* con el identificador del documento creado, y el resultado de esta llamada es lo que devuelve la función de *create().*

6.8.1.1.3 INSERT en la clase *Table*

El diagrama de la Ilustración 6-47 explica con detalle el funcionamiento de la función *INSERT*, que se utiliza dentro de la función *create()* para crear un documento.

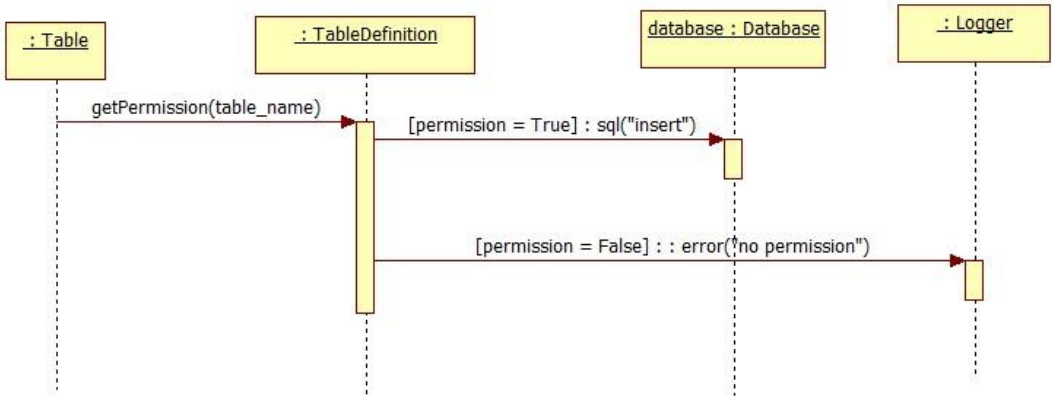

Ilustración 6-47 Diagrama de secuencia de *INSERT* en la clase *Table*

La figura anterior muestra como *Table* comprueba los permisos sobre la tabla en cuestión haciendo una llamada a *getPermission()* de *TableDefinition*. Si se tienen los permisos necesarios, se realiza una inserción en la instancia *database* de la clase *Database.* Si no se tienen los permisos, se registra el error "*no permission"* mediante la función *error()* de la clase *Logger.*

6.8.1.1.4 Borrar documento

Cuando el usuario selecciona borrar un documento en la interfaz gráfica, el cliente realiza una petición al servidor, que procesa esta petición y genera una llamada a la función *delete()* de la clase *Document.*

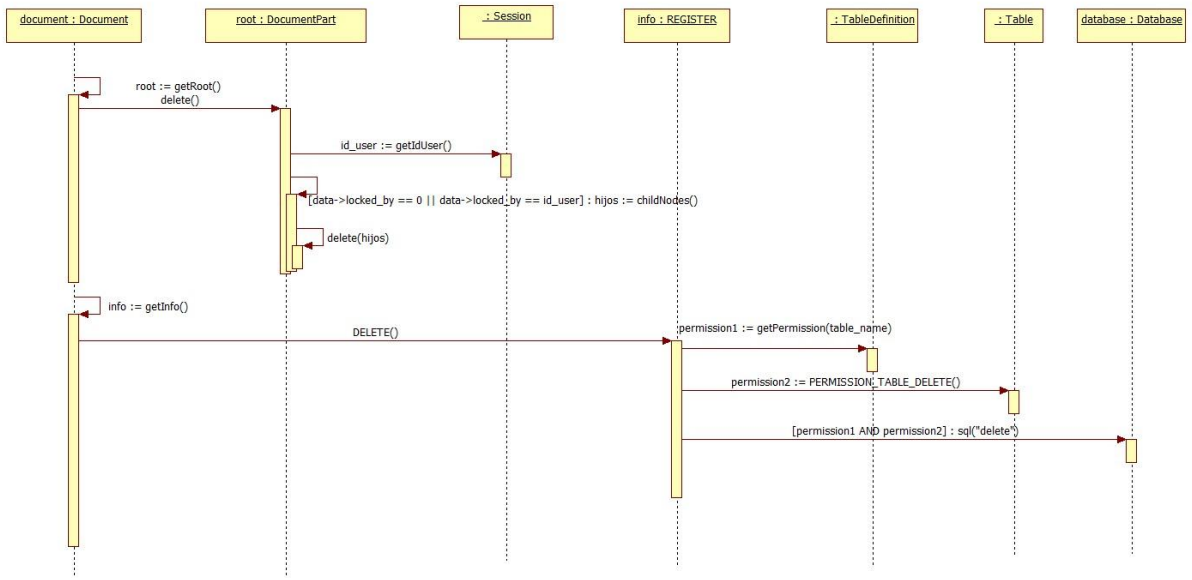

Ilustración 6-48 Diagrama de secuencia de borrar documento

En el diagrama de la Ilustración 6-48 muestra cómo se obtiene la raíz del documento con *getRoot().* Una vez obtenida la raíz (*root*) del documento, se llama a la función *delete()* de dicha raíz, que es de tipo *documentPart.* Esta función comprueba los permisos del usuario para realizar la operación de borrado. Si los permisos son correctos, la instancia de *Document* obtiene todos los hijos de la raíz y procede a su borrado recursivo. Finalmente borra la información del documento. Para simplificar el diagrama se ha extraído el desarrollo de la función *delete()* de *documentPart* y se ha incluido en un diagrama separado (ver Ilustración 6-49). En dicho diagrama se muestra el borrado de una parte después de haber comprobado los permisos y después de comprobar que la parte no es de tipo raíz. Las partes están relacionadas de manera que forman una lista enlazada para mantener el orden. Por ello, para borrar la parte, primero se busca el hermano anterior y se actualiza la tabla actualizando el siguiente, de manera que la lista continúa enlazada manteniendo el orden correcto. Finalmente se realiza la llamada a *DELETE* del registro de datos de la parte que se está borrando de la misma manera que en el diagrama de la Ilustración 6-48.

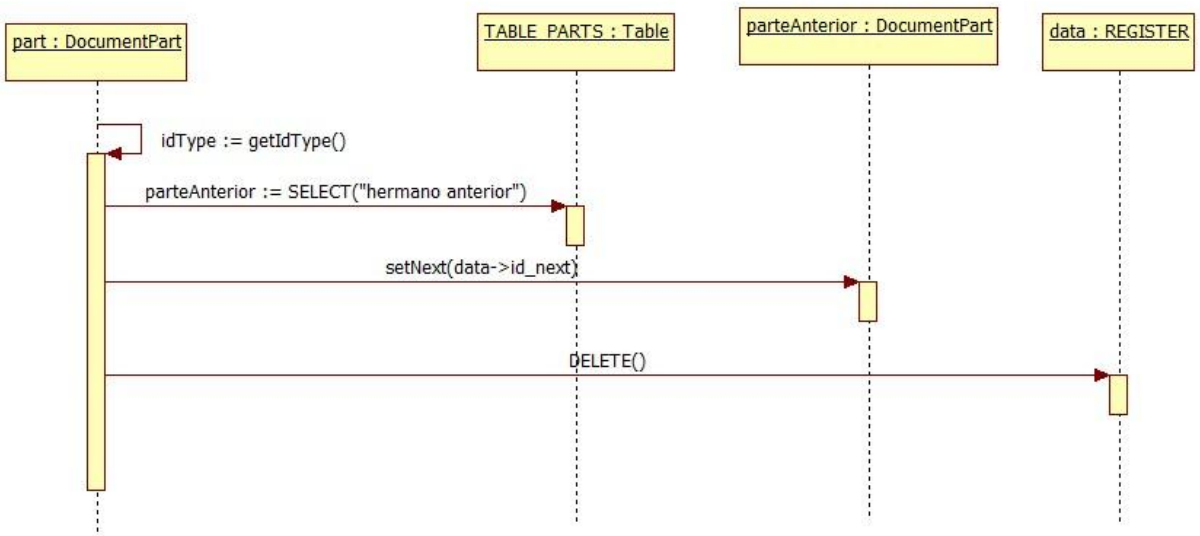

Ilustración 6-49 Función *delete*() de *documentPart*

6.8.1.1.5 Obtener información de un contacto

Cuando un usuario selecciona un contacto de la lista de contactos o mediante el buscador, puede visualizar la información de ese contacto. Para llevar a cabo esta tarea, se produce a través de la *API* una llamada a la función *getContactInfo(id_usuario)* de la clase *Contacts.* El diagrama describe el funcionamiento a partir de esta llamada. Se obtiene el usuario de la tabla de usuarios (mediante el *SELECT())* y en la condición del *SELECT* se indica que el identificador de usuario, sea el *id_usuario* que es parámetro de *getContactInfo(id_usuario).* Si el usuario en concreto existe, se devuelve la información de usuario, y si no, se devuelve la indicación de que no hay resultados con ese identificador (es decir, vacío).

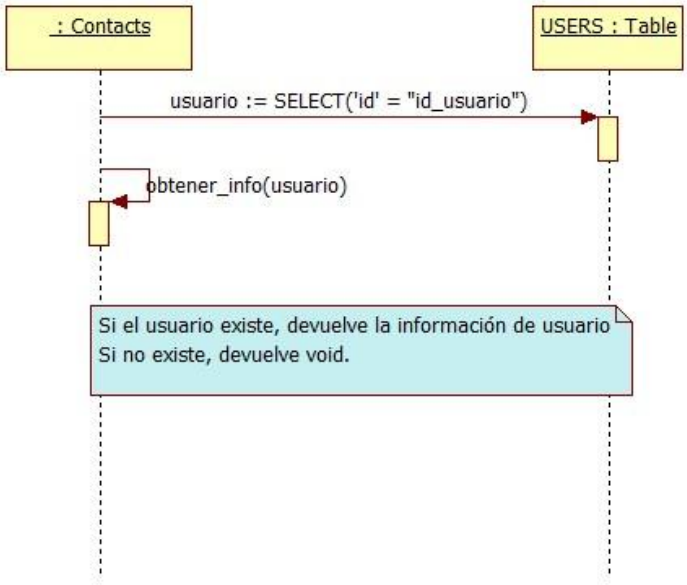

Ilustración 6-50 Diagrama de secuencia de obtener información de usuario

6.8.1.1.6 Añadir contacto a los contactos del usuario

El diagrama muestra la situación cuando se intenta añadir un contacto para que forme parte de los contactos del usuario. Si el usuario seleccionado ya fuera parte de ese grupo de contactos, no se añadiría de nuevo y no se produciría ningún cambio; si no lo es, se procede a añadirlo en las correspondientes estructuras. El proceso es como se explica a continuación.

La clase *Contacts* recibe una llamada a su función *addFriend(id_usuario)* con el identificador del usuario que se desea añadir. Lo primero que hace es comprobar que no sea ya amigo (contacto). Si ya lo es, no lleva a cabo ninguna acción. Si no lo es, procede a añadirlo insertando el registro en la tabla de contactos del usuario. Para comprobar si es amigo o no, realiza una llamada a *SELECT*() sobre la tabla de contactos, y en la condición del *SELECT* se establece que el *id_usuario* que se pasa como *addFriend* sea igual que el identificador de algún usuario de la lista de amigos (incluyendo al propio usuario). Los detalles de esta comprobación se han omitido en el diagrama, indicando únicamente la llamada a la función.

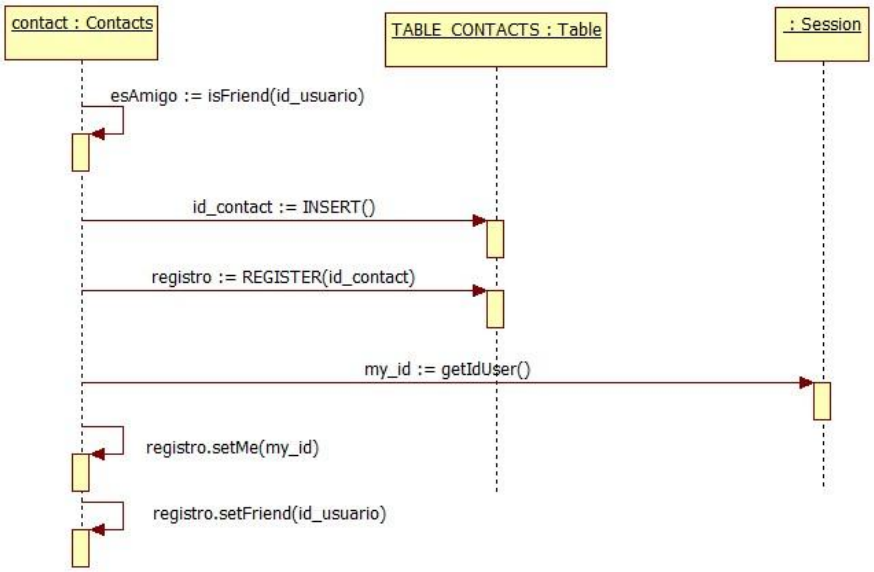

Ilustración 6-51 Diagrama de secuencia de añadir contacto a mis contactos

6.8.1.1.7 Borrar contacto de Mis contactos

Cuando se está visualizando la información de un contacto que forma parte de los contactos del usuario, se puede seleccionar la opción de "Borrar contacto". A continuación se explica cómo se realiza este borrado cuando la *API* llama a la función *removeFriend(id_contact)* con el identificador del contacto que se desea eliminar.

El diagrama muestra cómo se selecciona el contacto con el identificador igual al *id_contact* que se desea eliminar mediante una llamada a la función *SELECT* y con la condición mencionada. Una vez obtenido, se elimina el registro del contacto llamando a *DELETE*.

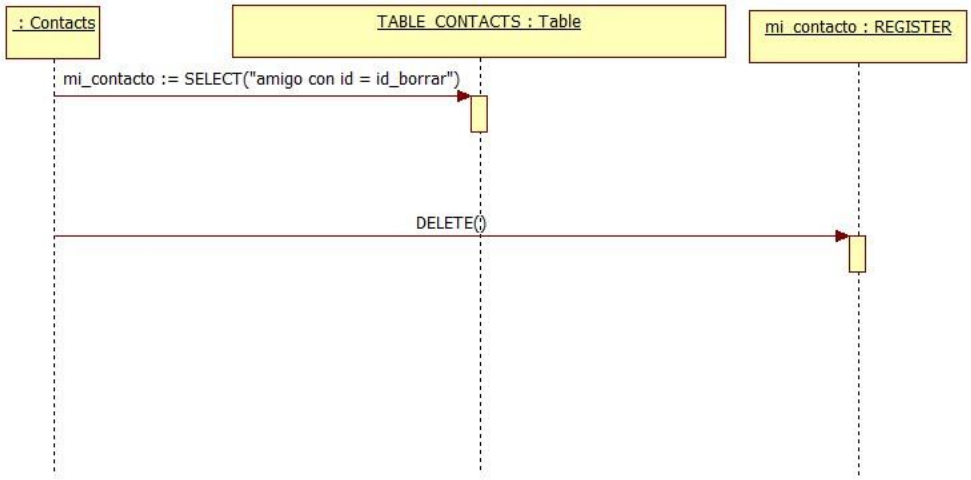

Ilustración 6-52 Diagrama de secuencia de borrar contacto

7 Ejemplo de uso del sistema

Para realizar cualquier acción, lo primero que debe realizar el usuario es introducir su dirección de usuario y contraseña en la pantalla de login.

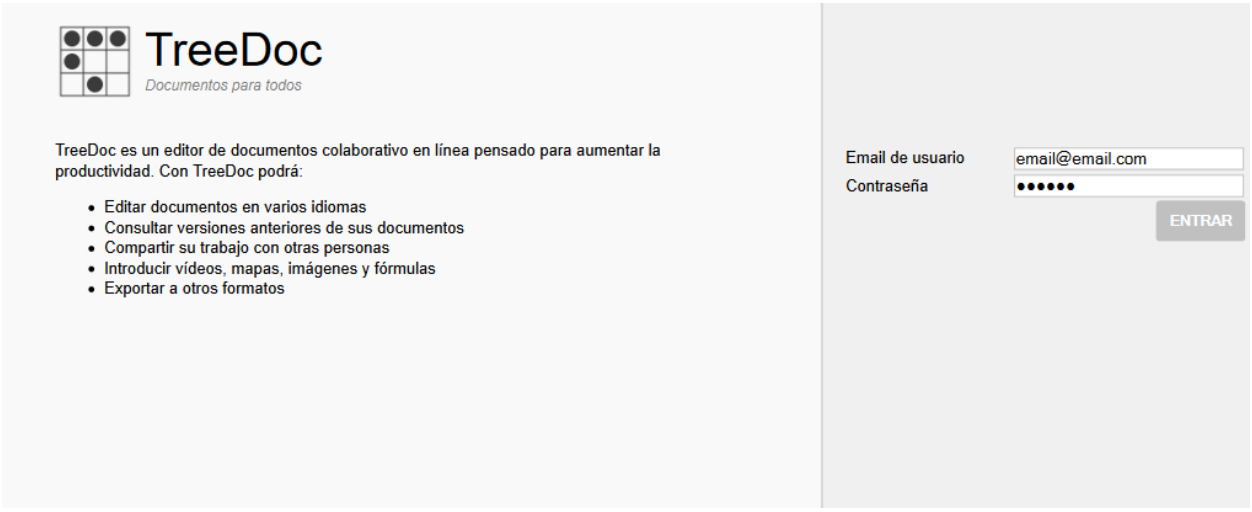

Ilustración 7-1 Pantalla de login

Una vez introducidos el nombre de usuario y contraseña, se pueden llevar a cabo el resto de acciones tras hacer clic en el botón "*ENTRAR".*

7.1 Crear un documento nuevo y editarlo

1. Acceder a la sección de "*Documentos".* Es la sección que aparece por defecto tras el login. Para acceder a esta sección en cualquier momento sólo es necesario seleccionar *Documentos* en la barra superior.
2. Seleccionar el botón "*CREAR DOCUMENTO".*

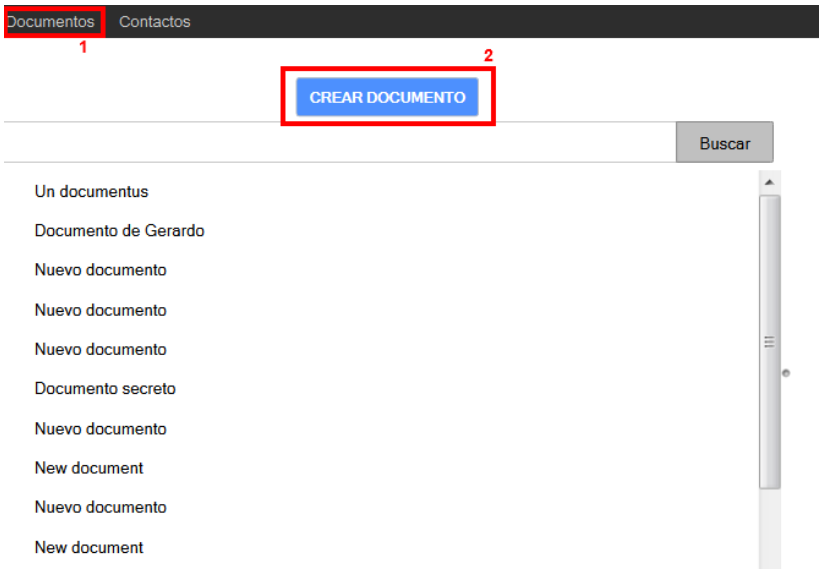

Ilustración 7-2 Crear documento nuevo

Se mostrará el editor de documentos y un documento en blanco.

3. Introducir la información básica del documento. Seleccionar el icono de *Configuración, y a* continuación Propiedades del documento e introducir el *título*, *descripción* y *keywords* del documento.

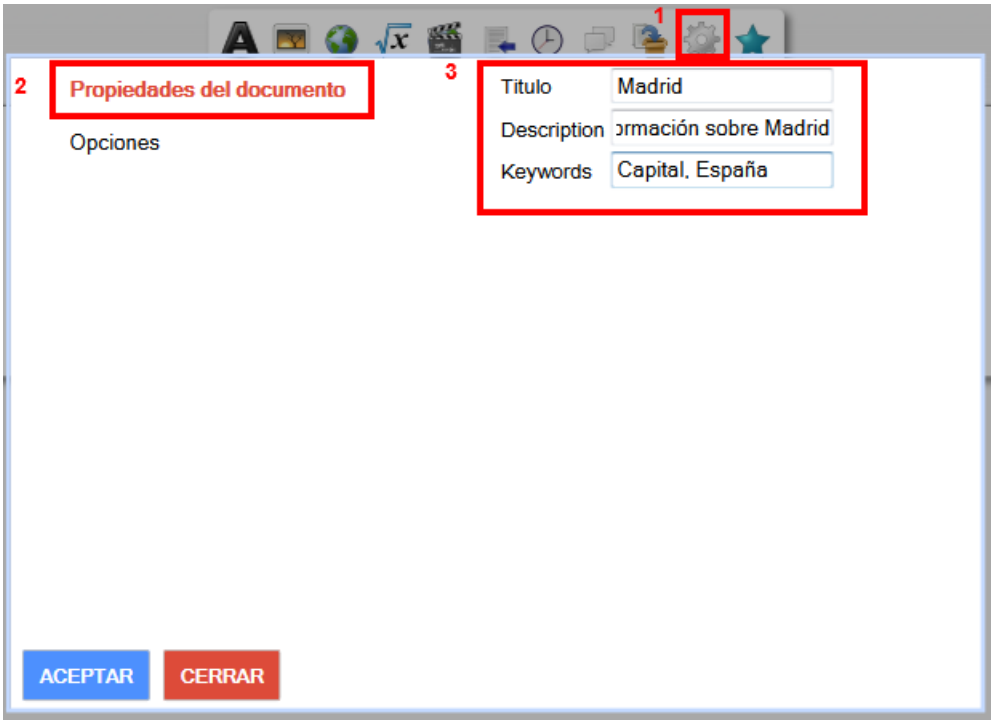

Ilustración 7-3 Propiedades del documento

Seleccionando el botón de *ACEPTAR* se regresa al editor, y se puede ver que el título cambia tras seleccionar el botón de *Actualizar* en el esquema del documento. (área 1 de la Ilustración 7-4.

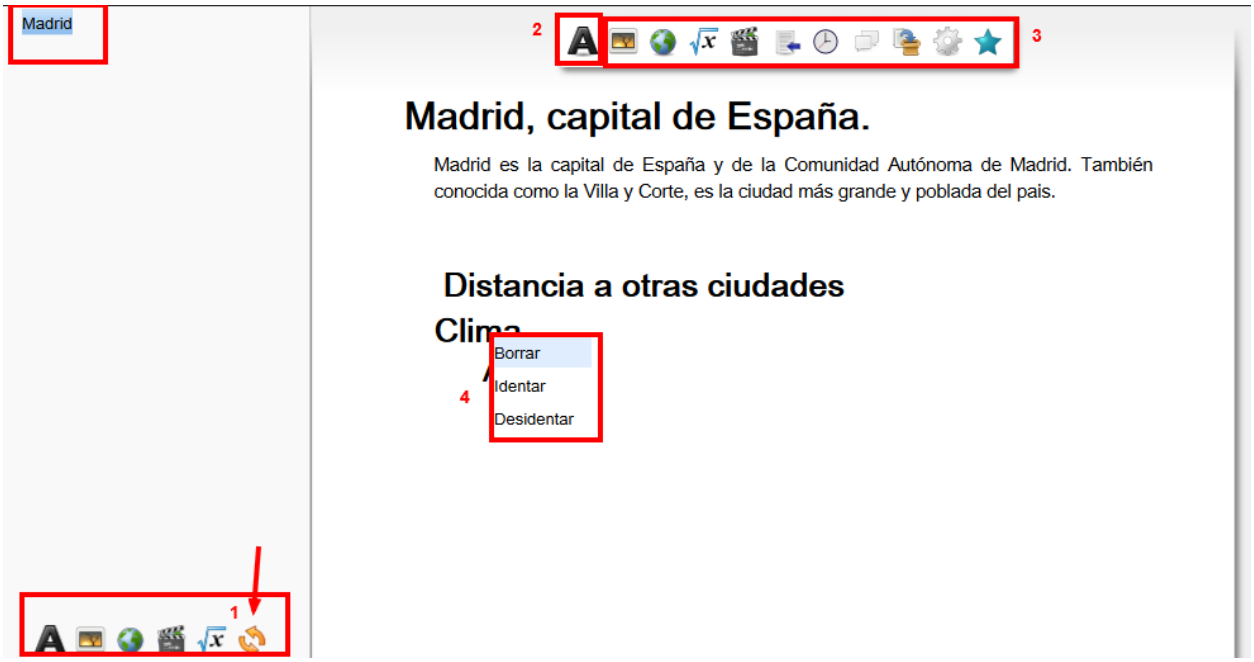

Ilustración 7-4 Edición de documento

4. Introducir una sección en el documento, seleccionando el icono de *Sección (*sección 2 de la Ilustración 7-4). Se puede escribir el texto del título de la sección y el texto de la sección.
5. Para insertar subsecciones se debe seleccionar de nuevo el botón de *Sección*. Para elegir el sangrado correcto se puede pulsar el botón derecho del ratón y seleccionar *Indentar y desindentar* para obtener la estructura de documento deseada. (Área 4 de la Ilustración 7-4).
6. Para introducir otro tipo de contenido dentro de una sección, hay que seleccionar el tipo en la barra superior de las herramientas del editor y posteriormente introducir los datos necesarios para la sección correspondiente. Ver la sección 6.2.5 para más información acerca de la inserción de los distintos tipos de partes.
7. Para borrar una parte del documento, se debe seleccionar la parte con el botón derecho del ratón, y posteriormente la opción *Borrar.* La parte seleccionada desaparecerá y las sub-secciones de esta (si las tuviera) también serán eliminadas.

8 Conclusiones

Este trabajo se ha centrado en realizar una herramienta colaborativa de gestión de documentos intuitiva, versátil y eficaz. Para ello se han analizado las características más relevantes de este tipo de sistemas y adoptado algunas de las propuestas que están teniendo más éxito en la actualidad en ellos. El resultado ha sido un entorno que permite colaborar simultáneamente (en tiempo real) a múltiples usuarios en el uso y elaboración de documentos capaces de albergar múltiples tipos de contenidos. Además incluye un sistema de gestión de documentos que permite asignar distintos permisos a los usuarios sobre los documentos, lo que permite diferenciar roles en el sistema.

La representación de los documentos se basa en una estructura de árbol donde las relaciones padre-hijo representan relaciones entre una parte del documento y las partes incluidas en él, y las relaciones entre nodos del árbol con un mismo padre representan la secuenciación de una serie de partes. La adopción de esta estructura facilita la gestión interna a la aplicación de ciertas modificaciones comunes del documento del documento (ej. añadir y borrar partes) y la propagación de propiedades y acciones a través del mismo (ej. cambios de permisos o formato). Además, se trata de un tipo de representación similar a estándares como XML, lo que facilita el uso de tecnologías asociados a ellos.

A fin de dotar de riqueza de medios a los usuarios, se han implementado distintos tipos de partes para el documento. Las disponibles actualmente son títulos, texto plano, ecuaciones, imágenes, vídeos, enlaces a otros documentos y comentarios. El diseño del sistema está preparado para incluir nuevos tipos de partes, como archivos de audio o código incrustado. Para ellos sólo habría que suministrar las clases que implementan las interfaces correspondientes a las partes de un documento y actualizar la base de datos.

El sistema también ofrece funcionalidad para la gestión de documentos. Para cada documento, a nivel de sus partes y para cada usuario se pueden establecer permisos de lectura, modificación, borrado y administración. Ello proporciona un mecanismo flexible para establecer diferentes roles de los usuarios respecto a los documentos.

También hay que resaltar entre las características del sistema el registro de actividad y el control de versiones. El sistema registra todos los cambios que se producen en los documentos. Esto permite mantener un historial de versiones de los documentos, tanto para consultar versiones pasadas como para analizar su evolución.

Todas estas características han permitido satisfacer los criterios iniciales de construir un sistema flexible en cuanto a contenidos, gestión de usuarios e integración con tecnologías de otras partes. Al tiempo, el sistema incorpora buena parte de las características disponibles en

sistemas similares disponibles en la actualidad, aunque con especial hincapié en la facilidad para modificar el sistema para abordar nuevas necesidades.

El desarrollo de TreeDoc se ha enfrentado a varios retos técnicos. Estos han tenido principalmente su origen en el rápido desarrollo y cambio de las tecnologías empleadas en la web y usadas en la aplicación, y en la heterogeneidad de sus implementaciones. En este aspecto, la parte que mayor reto ha supuesto ha sido sin duda la elaboración del editor de documentos.

La interfaz gráfica del editor se ha desarrollado utilizado GWT, un *framework* que en general acelera el desarrollo de aplicaciones RIA. En determinadas partes críticas ha sido indispensable un grado de conocimiento muy alto de CSS y de la API de *JavaScript*. Durante el tiempo del proyecto GWT ha cambiado de versión en tres ocasiones. Por otra parte, los navegadores usados para acceder a la aplicación han estado en permanente evolución: FireFox ha pasado por varias versiones (cuando se comenzó a desarrollar el sistema, se encontraba en la versión 3.6); el navegador Google Chrome también ha cambiado de versión en varias ocasiones (Se comenzó con su versión 9). Las nuevas versiones han implementado nuevas características de CSS, obligando a revisar y reescribir partes del código relacionado con CSS. En general, ha sido necesario un continuo esfuerzo de adaptación del desarrollo a los cambios en las tecnologías punteras en el ámbito del desarrollo web.

En relación con la variedad de navegadores y los cambios en sus versiones, hay que señalar también las complicaciones surgidas de sus variaciones en la implementación de los estándares. Los diferentes navegadores (ej. Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, Opera o Safari) incluyen implementaciones con variantes propietarias de los estándares marcados por la W3C (HTML y CSS). Por ejemplo, hay variaciones a la hora de mostrar las listas, o el comportamiento por defecto de los botones sin establecer los valores de algunos atributos puede ser diferente.

Respecto al proceso de desarrollo en sí, hay que señalar que se partió de un estado inicial con numerosas alternativas tecnológicas y de diseño. Para explorarlas adecuadamente se decidió adoptar un enfoque ágil con un uso intensivo de prototipos. La herramienta final es fruto de la integración y adaptación de las alternativas que se mostraron más apropiadas.

9 Trabajo futuro

En este apartado recoge algunas ideas para futuros trabajos en la línea de este proyecto. Se trata de potenciales expansiones en la línea de características disponibles en algunas herramientas de edición colaborativa o que los usuarios empiezan a requerir.

En primer lugar, se podría implementar tolerancia a fallos ante la pérdida de la plataforma, lo que proporcionaría mayor disponibilidad de los documentos. Esto se podría llevar a cabo por medio del almacenamiento distribuido. Habría que tener en cuenta aspectos avanzados de replicado y control de la concurrencia entre otros, pero se conseguiría una aplicación más robusta.

También sería interesante implementar de alguna manera la funcionalidad de flujos de trabajo. Se trata de definir los pasos que deben seguir los usuarios para elaborar un documento completamente: qué usuarios deben elaborar cada parte del documento y en qué orden. Una aplicación como la que se ha desarrollado que incluyera esta funcionalidad sería de gran utilidad para entornos de trabajo con procesos bien definidos como la administración pública, grandes empresas y organizaciones sujetas a estándares de calidad.

En la línea de lo anterior aparece la firma digital. La posibilidad de firmar un documento digitalmente completaría la funcionalidad de flujos de trabajo. Sería útil en organizaciones donde algunos documentos tienen que seguir un ciclo predefinido de elaboración y revisión de sus partes para llegar al usuario final. Por ejemplo, un usuario rellena una serie de campos, otro usuario lo firma digitalmente con el sello de un departamento para confirmar que todo es correcto, y un usuario final lo firma y envía al interesado.

Finalmente, se podría considerar evaluar el aumento o disminución de productividad que se obtiene utilizando este sistema para la edición colaborativa en un entorno real. La aplicación desarrollada se puede usar en escenarios muy diferentes, por ejemplo empresas que generan muchos documentos que deben editar varios usuarios o entornos de aprendizaje (plataformas *e-learning*). Los requisitos de estos entornos son muy diferentes, por lo que se podría considerar qué tipos de características en los sistemas de edición colaborativa son más adecuados para cada uno de ellos y en qué medida afectan a la productividad. Algunos factores a considerar serían el tiempo que lleva a un usuario aprender a manejar la aplicación, y la rapidez con que puede llevar a cabo su trabajo comparando con la situación actual u otros sistemas.

10 Referencias

1. Sitio web de Drupal. [En línea] [Citado el: 11 de Agosto de 2011.] http://drupal.org/.

2. Sitio web w3c XML. [En línea] [Citado el: 11 de Agosto de 2011.] http://www.w3.org/XML/.

3. Sitio web W3 CSS. [En línea] [Citado el: 29 de Agosto de 2011.] http://www.w3.org/Style/CSS/.

4. **Chaffey, Dave.** *Groupware, Workflow and Intranets - Reenginering the Enterprise with Collaborative Software.* s.l. : Digital Press, 1998.

5. **Canseco, V. y Gerónimo, G.** *Breve introducción a los sistemas colaborativos: Groupware & Workflow.* 1998.

6. Sitio Web de Gartner. [En línea] [Citado el: 25 de Agosto de 2011.] http://www.gartner.com/technology/research/.

7. **NIST.** Definición y características de Cloud Computing. [En línea] Enero de 2011. [Citado el: 12 de Agosto de 2011.] http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

8. Sitio web de Google Maps. [En línea] [Citado el: 29 de Agosto de 2011.] http://maps.google.com.

9. Sitio Web de LaTex. [En línea] [Citado el: 28 de Agosto de 2011.] http://www.latex-project.org/.

10. Sitio Web de Youtube. [En línea] [Citado el: 27 de Agosto de 2011.] http://www.youtube.com/.

11. **Gamma, Erich, y otros.** *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley, 1995.

12. Sitio Web de Google GWT. [En línea] [Citado el: 9 de Agosto de 2011.] http://code.google.com/webtoolkit/.

13. Sitio Web de GNU. [En línea] [Citado el: 1 de Septiembre de 2011.] http://www.gnu.org/copyleft/gpl.html.

14. Sitio Web de MS Office. [En línea] [Citado el: 22 de Agosto de 2011.] http://office.microsoft.com/es-es/word/caracteristicas-y-ventajas-de-word-2010-HA101810003.aspx.

15. Sitio web de Gobby. [En línea] [Citado el: 22 de Agosto de 2011.] http://gobby.0x539.de/trac/.

16. Sitio web de AbiWord. [En línea] [Citado el: 1 de Agosto de 2011.] http://abisource.com/.

17. Sitio web de Synchroedit. [En línea] [Citado el: 29 de Agosto de 2011.] http://www.synchroedit.com/.

18. Sitio web de MobWrite. [En línea] [Citado el: 21 de Agosto de 2011.] http://www.mobwrite.net/.

19. Sitio web de Google Docs. [En línea] [Citado el: 20 de Agosto de 2011.] http://docs.google.com/support/bin/answer.py?hl=en&answer=49008.

20. Sitio Web de GWT. [En línea] [Citado el: 22 de Agosto de 2011.] http://code.google.com/webtoolkit/.

21. Sitio web de Eclipse. [En línea] [Citado el: 28 de Agosto de 2011.] http://www.eclipse.org/.

22. Sitio web de FireFTP. [En línea] [Citado el: 20 de Agosto de 2011.] https://addons.mozilla.org/es-es/firefox/addon/fireftp/.

23. Sitio web de Firebug. [En línea] [Citado el: 29 de Agosto de 2011.] https://addons.mozilla.org/es-es/firefox/addon/firebug/.

24. Sitio web de HttpFox. [En línea] [Citado el: 29 de Agosto de 2011.] https://addons.mozilla.org/es-es/firefox/addon/httpfox.

25. Sitio Web de XDebug. [En línea] [Citado el: 28 de Agosto de 2011.] http://xdebug.org/.

26. Sitio Web de InnoDB. [En línea] [Citado el: 27 de Agosto de 2011.] http://www.innodb.com/.

27. Sitio web de comparativa de motores de referencia. [En línea] http://www.treeweb.es/TreeWeb/Articulos/MySQL/Motores-de-almacenamiento.

28. Modelo de permisos extendido con grupos. [En línea] [Citado el: 22 de Agosto de 2011.] http://www.treeweb.es/Proyectos/TreeWeb/R4/Investigacion/002-Modelo-de-permisos-extendido-con-grupos.

29. Sitio web de la nueva interfaz gráfica en JavaScript nativo. [En línea] [Citado el: 29 de Agosto de 2011.] http://www.treeweb.es/ShareCode/c769ac6201aa4cb87503413804a1d7c4/.

30. Sitio web del prototipo de redimensionado de imágenes de la interfaz. [En línea] [Citado el: 30 de Agosto de 2011.] http://www.treeweb.es/ShareCode/5bc2dfb963ff876854d829f4c7d96176.

31. Sitio web del prototipo de carga dinámica de un documento. [En línea] [Citado el: 30 de Agosto de 2011.] http://www.treeweb.es/ShareCode/e87e69e6b221f824765353c36855cf41/.

32. Ejemplo de comparación de creación de tabla con prototipo y con la última versión. [En línea] [Citado el: 30 de Agosto de 2011.] http://www.treeweb.es/ShareCode/0c6fe30eddb63d949d8680f43d57f5c0/.

33. Sitio Web de InnoDB. [En línea] http://www.innodb.com/.

11 Glosario

- **AJAX:** JavaScript asíncrono y XML (*Asynchronous JavaScript And XML*), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el navegador de los usuarios, y al mismo tiempo se mantiene la comunicación asíncrona con el servidor en segundo plano. Esto permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.
- ***Awareness*:** Mecanismos que permiten que un usuario de un entorno colaborativo perciba la actividad que están llevando a cabo el resto de usuarios.
- **API:** Interfaz de programación de aplicaciones. Proporciona un conjunto de funciones para facilitar la comunicación entre dos componentes.
- **Blog:** Sitio web que permite al autor/ autores escribir artículos periódicamente. El sitio muestra estos artículos ordenados cronológicamente
- **Cloud Computing:** (Definición del NIST) modelo para habilitar acceso conveniente por demanda a un conjunto compartido de recursos computacionales configurables, por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios, que pueden ser rápidamente aprovisionados y liberados con un esfuerzo mínimo de administración o de interacción con el proveedor de servicios. Este modelo de nube promueve la disponibilidad y está compuesto por cinco características esenciales, tres modelos de servicio y cuatro modelos de despliegue.(7)
- **CMS:** Sistema de gestión de contenidos (Content Management System). Sistema que permite administrar contenidos de distintos tipos en un medio digital.
- **CSS:** *Cascading Style Sheet.* Mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. (3)
- **Documento:** Un documento es un conjunto de entidades del tipo *Parte* que se ordenan y se estructuran de forma jerárquica. La longitud de un documento puede ser desde un simple párrafo hasta un libro completo de varios volúmenes impresos.
- **DOM:** *Document Object Model* o Modelo de Objetos del Documento. Es una *API* que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos. A través del *DOM*, los programas pueden acceder y modificar el contenido, estructura y estilo de los documentos *HTML* y *XML*, que es para lo que se diseñó principalmente. Su responsable es el World Wide Web Consortium (W3C).
- **e-book**: Libro electrónico.

- **Editor de texto colaborativo:** Aplicación de Softare colaborativo que permite crear documentos y editarlos de forma colaborativa. Es decir, varios usuarios intervienen en un mismo documento.
- **Fórmula LaTex:** Fórmula matemática escrita utilizando el lenguaje de marcado "LaTex".
- **Front-end:** Parte de la aplicación que interactúa con el usuario.
- **GNU/GPL:** GNU *General Public License*. Licencia libre, copyleft para software y otros tipos de trabajo. (13)
- **GUI:** *Graphic User Interface,* interfaz gráfica de usuario.
- **HTML:** (*HyperText Markup Language* -Lenguaje de Marcado de Hipertexto).
- **IDE:** (Integrated development enviroment). Aplicaciones informáticas que proporcionan un entorno de trabajo que facilite el trabajo a los desarrolladores de *software*.
- **IDPF:** *International Digital Publishing Forum*.
- **JavaScript**: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
  Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente *widgets*) es también significativo. [Wikipedia]
- **Keywords**: Palabras clave. En el caso de un gestor de documentos, palabras clave relacionadas con el documento que permiten hacerse una idea del contenido tratado en el mismo y encontrar el documento fácilmente.
- **NIST**:*National Institute of Standards and Technology.*
- **Parte:** Es cada uno de los elementos que forman un documento. Hay distintos tipos: título, texto, tabla, gráfico, mapa, vídeo, lista, etc. Todos ellos tienen operaciones comunes y operaciones propias.
- **PDF:** *Portable Document Format.*
- **Plugin**: Módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.
- **Publicacion:** Una publicación consiste en la tupla que define un instante de tiempo en el que un determinado Documento está terminado o se haya decidido que se ha producido un cambio importante. Cada documento tiene una lista de instantes de tiempo que definen sus publicaciones.
- **RIA**: *Rich Internet Application.*
- **SGBDR**: Sistema gestor base de datos. Software que sirve de interfaz entre la base de datos, el usuario, y las aplicaciones utilizadas.
- **Software colaborativo o *Groupware*:** Conjunto de programas informáticos que integran el trabajo en un sólo proyecto con muchos usuarios concurrentes que se encuentran en diversas estaciones de trabajo, conectadas a través de una red (internet o intranet). [Wikipedia].

- **SQL:** (*Structured Query Language*)
- **TDF:** TreeDoc Format.
- **URL:** *Uniform resource locator*. Es una dirección única para un recurso que está en internet.
- **Widgets**: Pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de *widgets* o *Widget Engine*. Entre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual. Sin embargo, los *widgets* pueden hacer todo lo que la imaginación desee e interactuar con servicios e información distribuida en Internet. [Wikipedia]
- **Wiki**: Sitio web donde los usuarios vuelcan su conocimiento y el resto de usuarios puede editarlo, completarlo y corregirlo.
- **XML**: Extensible Markup Language (2).

12 Apéndice A: Sistemas de edición de texto

Este apartado presenta información sobre los sistemas descritos en el estado del arte. Se dividen entre aplicaciones de escritorio y aplicaciones web.

12.1 Aplicaciones de escritorio

- Microsoft Word(14): Es el editor más utilizado. En la versión 2010 permite crear documentos colaborativos y almacenarlos online para compartirlos. En realidad está dentro de los dos grupos (aplicaciones de escritorio y basadas en navegador) ya que también permite acceder a los documentos a través del navegador y realizar algunas ediciones.
- Gobby(15): Proyecto de código libre. Es un editor colaborativo que soporta edición de varios documentos en una sesión y un chat multi usuario para la comunicación entre los usuarios. Es multiplataforma y está disponible para Microsoft Windows, Mac OS X, y Linux.
- Abiword (16): Procesador de texto que permite colaborar en tiempo real en el mismo documento. Con el plug-in AbiCollab permite almacenar documentos online, compartirlos, y convertir a varios formatos.

12.2 Editores colaborativos basados en navegador web

- *Writely*: Fue uno de los editores basados en navegador web que más rápido creció. Google lo compró en 2006 y lo convirtió en "*Google Docs & Spreadsheets*".
- Synchroedit(17) y MobWrite (18): Dos ejemplos más de editores basados en navegador Web. Los dos tenían como fin conseguir una colaboración en tiempo real, aunque en documentos de gran tamaño, no se consigue.
- *Google Wave* fue un entorno de colaboración de Google. En 2010 dejó de desarrollarse como un proyecto independiente por no contar con suficiente aceptación por parte de los usuarios. Incorporaba la colaboración verdaderamente en tiempo real, que se incluyó en *Google Docs* (explicado posteriormente) más tarde.
- *EtherPad* fue uno de los primeros editores web en proporcionar colaboración en tiempo real carácter a carácter. Esto significa que cuando un usuario modifica un carácter, el

resto de usuarios que están visualizando el documento ve el cambio en ese carácter inmediatamente. Se puede descargar para instalar en un servidor o usar en algún sitio público. Finalmente fue comprado por Google y el equipo de desarrollo trabajó en el proyecto de Google wave.

- *Google Docs* es una aplicación web que incluye un editor de texto. Este editor permite importar documentos de distintos formatos y realizar conversiones. Se puede colaborar en tiempo real e incluye un chat para hablar con el resto de colaboradores. También proporciona los historiales de revisión de los documentos y permite volver a versiones anteriores de los mismos. Para ampliar información sobre Google Docs, se puede consultar su sitio web (19).

13 Apéndice B: Tecnologías y herramientas utilizadas

13.1 GWT

Una de las principales características del sistema es que proporciona colaboración en tiempo real. Para ello es necesario el uso de AJAX ya que de esta manera se consigue una comunicación entre cliente y servidor que permite modificar la página actual sin necesidad de refresco y obtener aplicaciones web totalmente dinámicas

GWT (*Google Web Toolkit)* facilita las funcionalidades de AJAX y este fue uno de los principales motivos para su elección.El desarrollo de las interfaces gráfica para la gestión de documentos, contactos y el área de edición se ha llevado a cabo utilizando GWT. Es la principal novedad tecnológica usada en este proyecto. Se trata de un *framework* desarrollado por *Google Inc.* cuya principal utilidad es la de traducir del lenguaje *Java* a *Javascript* y facilitar las funcionalidades de AJAX. Su primera versión apareció a mediados de 2006. Y hasta la fecha ha presentado gran número de versiones.

Otros motivos que ayudaron a su elección es que su utilización es gratuita y que existen diversos *plugins* que se integran con los principales *IDEs* de programación tales como *Netbeans* o Eclipse, sirven para tener un manejo más cómodo de las herramientas de GWT, tales como compilación y desplegado de la aplicación. *GWT* tiene dos modos de ejecución *Hosted* y *Web mode*, el primero para depurar y el segundo para traducción de código.

Referente a la interfaz gráfica, *GWT* ofrece distintos componentes para realizar los paneles de usuario, tales como botones, campos de texto, menús desplegables, etc., muy similares a *SWING* de Java. Además, cuenta con diversas librerías para agregar otros componentes y/o funcionalidades extra. De este proyecto han surgido proyectos similares como puede ser *SmartGWT*, que contiene componentes más avanzados y visualmente más atractivos.

Para más información sobre *GWT,* consultar su sitio web (20).

13.2 Apache

Debido a la elección de PHP, prácticamente es obligatorio contar con el servidor Apache para la interpretación de código. Aparte de ello, la potencia del servidor, la documentación, el respaldo técnico y el conocimiento previo de esta herramienta han apoyado su elección frente a otras alternativas.

Apache es un servidor HTTP de código abierto multiplataforma que implementa el protocolo HTTP/1.12. Su primera versión se publicó en 1996, escrito en C, altamente configurable, a día de hoy se encuentra en la versión 2.2.17.

Apache cuenta con multitud de módulos que permiten extender la funcionalidad del servidor, tales como autenticación, interpretación de código (PHP, Perl, Phyton), balance de carga, reescritura de URL, sitios virtuales, etc.

A día de hoy, es el servidor con mayor cuota de mercado (70%) y está empleado en la gran mayoría de sitios web del mundo, contando con el módulo de PHP para aplicaciones dinámicas. Su competencia más directa son los servidores *Apache Tomcat, JBoss* y *WebSphere*, todos ellos para Java, y el servidor IIS de *Microsoft* para ASP.

13.3 PHP

PHP (PHP Hyper-Text Preprocessor) es un lenguaje de programación semi-interpretado, diseñado originalmente para la creación de páginas Web dinámicas.

Es usado principalmente en interpretación del lado del servidor. El lenguaje PHP surgió en 1995 y a fecha de hoy se encuentra en su sexta versión.

Las principales ventajas de *PHP* son la portabilidad de código (multiplataforma), ser orientado a Web, su fácil conexión con bases de datos (*MySQL*, *PostgreSQL*, *Oracle*, *Microsoft SQL Server*, entre otras), la amplia documentación y ejemplos disponibles, y ser libre y gratuito. Estas ventajas, junto con el conocimiento previo del mismo por parte del equipo de desarrollo, han sido determinantes en su elección para la implementación de la lógica en el servidor de *TreeDoc*.

La integración de PHP con el servidor *Apache* es muy sencilla. Aunque es posible crear código PHP con cualquier editor de texto, lo más recomendable es utilizar alguna de estas herramientas que tienen resaltado de sintaxis y facilidades para escribir código, en este orden de preferencia: *Komodo Edit, Eclipse o Zend Framework, Netbeans, Gedit y NotePad++.*

PHP cuenta con multitud de librerías ya compiladas, aparte de este hecho y sirviendo como referente, sitios como *Facebook (*https://www.facebook.com/ *Wikipedia (*http://www.wikipedia.org/*)* o PHPMyAdmin (http://www.phpmyadmin.net/), están implementados con tecnología *PHP*.

13.4Eclipse, FireFTP, FireBug, HttpFox, xDebug

Estas son algunas de las herramientas que se han utilizado de forma directa en la implementación del código.

Eclipse es un entorno de programación, ofrece soporte para una gran cantidad de lenguajes además de una amplia gama de *plugins* como conexión y gestión de bases de datos, integración con *SVN*, integración con el servidor *Tomcat*, *JBoss*, etc.

Para más información, consultar el sitio Web de Eclipse (21).

FireFTP es un *plugin* del navegador *Firefox* destinada a gestionar cuentas FTP en el navegador, accede al disco duro del usuario y al sitio FTP y se pueden realizar operaciones de ficheros.

Para más información, consultar el sitio Web de *FireFTP* (22).

FireBug es un *plugin* para el navegador *Firefox* el cual permite realizar depuración de forma más amigable del código en el propio navegador, ofrece acceso a la consola de *JavaScript* y su depuración, también permite la inspección de elementos de HTML y su modificación entre otras funciones.

Para más información, consultar el sitio Web de *FireBug* (23).

HttpFox es un plugin para el navegador Firefox cuya funcionalidad principal es mostrar el tráfico existente en el navegador y visualizar los parámetros de las peticiones, asi como las respuestas del servidor.

Para más información, consultar el sitio web de HttpFox(24).

XDebug es una extensión de PHP que permite depurar y realizar *profiling*, la depuración tradicional en PHP es la salida con “echo” y el uso de *loggers*.

Para más información, consultar el sitio Web de *xDebug* (25).

13.5 MySQL

MySQL es un sistema Gestor de Bases de Datos Relacionales y es el utilizado para la persistencia de datos en *TreeDoc*.

Creado en 1995, y actualmente en su quinta versión, cumple el estándar *SQL*. Se ofrece bajo licencia *GNU GPL* salvo para empresas que lo incorporan a productos privativos. Es ampliamente usado en aplicaciones web y en multitud de aplicaciones, la gratuidad, facilidad de uso y el uso casi-transparente por parte de PHP han sido los principales factores en la decisión del uso de esta tecnología.

Por desgracia, MySQL no es la base de datos perfecta, al menos el uso del motor de búsqueda no transaccional MyISAM que puede provocar fallos de integridad y/o de datos debido a la concurrencia del uso en los documentos, por lo que es necesario implementar dicha lógica en PHP, por otra parte, el motor es ideal para la principal funcionalidad de la web que es la lectura de datos. Este motor puede provocar fallos de integridad y/o de datos debido a la concurrencia del uso en los documentos, por lo que es necesario implementar dicha lógica en PHP.

Sin embargo, de entre los motores disponibles, es el que mayor rendimiento proporciona debido a su gran velocidad en lecturas. El gráfico de la Ilustración 13-1 muestra una comparación del rendimiento entre distintos motores:

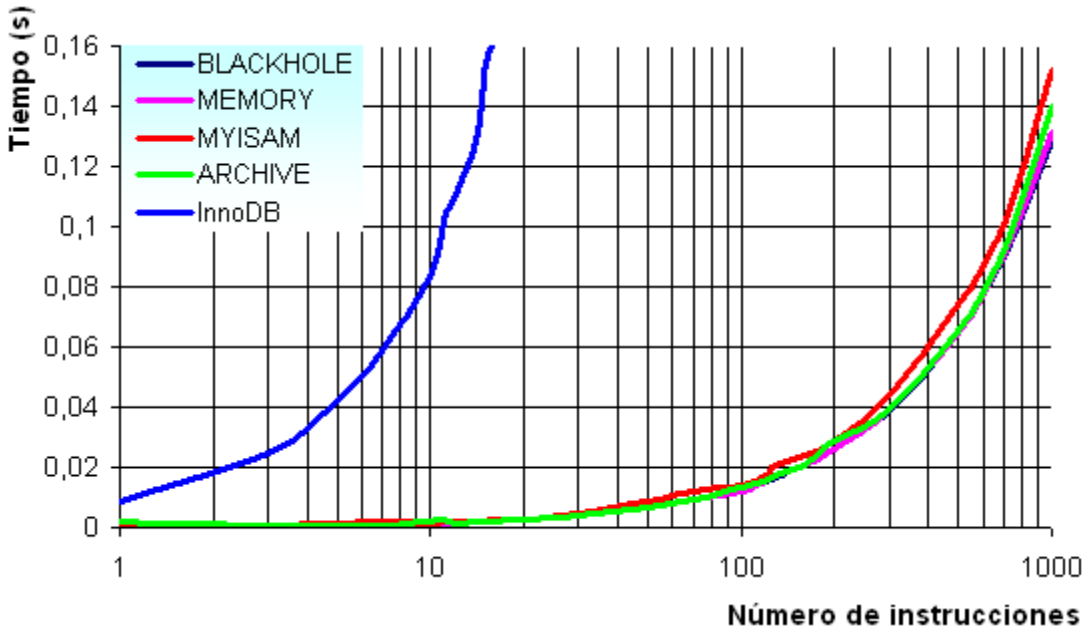

Ilustración 13-1 Comparación del rendimiento de los distintos motores

La Ilustración 13-1 Comparación del rendimiento de los distintos motores muestra en el eje vertical el tiempo que se tarda en completar una operación de inserción. Se puede ver que InnoDB (26) tiene un rendimiento muy bajo comparado con el resto, y que los demás obtienen valores bastante cercanos. Para más información sobre los motores de almacenamiento se pueden consultar las referencias(27).

13.6 Ubuntu Server

Ubuntu es una distribución de *Linux* basada en la distribución *Debian* cuya interfaz y uso es más atractiva y fácil frente a las anteriores distribuciones de Linux (*Debian, Gentoo, SuSe, Slackware, FreeBSD,* entre otras), gracias a estar basado en *Debian*, los paquetes de software son compatibles.

Ubuntu es libre y de código abierto e impulsado principalmente por la empresa *Canonical Ltd.* Existen varias distribuciones Ubuntu (*Desktop, Server,Kubuntu, Xubuntu*). La edición "server" está orientada a servidores y que cuentan con las principales herramientas típicas pre-instaladas tales como servidor web, SSH, FTP, etc., mientras que la distribución Desktop está orientada a usuarios de escritorio.

La decisión de usar Ubuntu Server frente a otras distribuciones de Linux se debe a la amplia documentación, facilidad de uso, conocimiento previo y el respaldo técnico existente, por otra parte, con respecto a utilizar otras tecnologías como Windows Server de Microsoft u OSX de Apple ha sido la gratuidad, eficiencia de los servidores Linux, el gran mercado del que disponen y la compatibilidad con multitud de arquitecturas hardware.

A día de hoy, Linux lidera el sector de los servidores y su situación es creciente tanto en empresas como en Administraciones Públicas, también se está incrementando su uso en los usuarios de escritorio.

13.7 SVN

Subversion es un sistema de control de versiones de código, también conocido por sus siglas (*SVN*), es la evolución del antiguo *CVS* con diversas mejoras, como agregar un identificador de versión a todo el repositorio además de no existir los riesgos de corrupción del repositorio a nivel de archivo. También permite la creación de *branches*, que equivalen a desarrollos alternativos de la aplicación a partir de un instante de tiempo.

SVN es software libre con licencia *Apache*, el uso del repositorio no es trivial e induce a errores; a pesar de ello se ha elegido como opción debido al desconocimiento de uso de otras tecnologías de control de versiones como pueden ser *Git* y *Mercurial*.

Para interactuar con el servidor *Subversion* existen distintas alternativas. Muchos IDEs de desarrollo como por ejemplo *Eclipse* y *Netbeans*, incorporan un cliente SVN para interactuar directamente con el repositorio y trabajar siempre con código actualizado.

Otra alternativa, bastante cómoda es utilizar el cliente TortoiseSVN (o su equivalente para sistemas *Linux RabbitVCS*) implementado como una extensión en el explorador de *Windows*, (*Nautilus* en Linux) Es software libre liberado bajo la licencia *GNU* y *GPL*, puede usarse sin entorno de desarrollo, simplemente seleccionado una carpeta, introduciendo los datos del repositorio y especificando el proyecto, la carpeta se actualiza acorde a la última versión. Su página oficial en la que se puede encontrar documentación y su descarga es http://tortoisesvn.net/. (http://www.rabbitvcs.org/ para RabbitVCS)

13.8 Java y Apache Tomcat

Java es un lenguaje de programación orientado a objetos, su sintaxis es muy similar a C++ y posee muchas de sus características como la orientación objetos, aunque es mucho más amigable que este ya que el modelo de objetos y la gestión de memoria está gestionada por un recolector de basura, el cual reduce casi por completo el problema de la gestión de memoria.

Java fue desarrollado por *Sun Microsystems* (en la actualidad *Oracle*) a principios de los 90, a día de hoy se encuentra en la versión 6, el código se ejecuta, etc, etc.

En un principio, el lenguaje Java no estaba orientado a la web (a diferencia de PHP), por lo que fue necesario la elaboración de servidores especializados que ejecutasen código Java, etc, etc, comenta CGI, etc. Algunos de los servidores que soporta la ejecución de Java en web son WebSphere,JBOSS y Tomcat

El **servidor Apache Tomcat** es un servidor web que soporta *servlets* y *JSP*s , de esa forma es posible utilizar la implementación realizada en Java para exportar los documentos.

Este servidor se ha utilizado para el módulo de exportación de *TreeDoc*. El motivo que llevó a ello en lugar de utilizar *Apache y PHP* es la falta de librerías adecuadas para exportar el formato ODT de *OpenOffice* con la segunda opción. También se reutilizó la plataforma para obtener los ficheros en formato PDF y EPUB.

14 Apéndice B: Modelo de permisos extendido con grupos

14.1 Descripción

Este modelo extiende la funcionalidad de los permisos de usuarios a grupos para facilitar la administración de permisos tanto en el aspecto técnico como en aspecto del usuario. Los cambios respecto de la versión anterior consisten en agregar una tabla común de grupos 'GRUPOS' y una tabla común que relacione los usuarios que estén en un grupo 'USU_GRU'. Además, por cada tabla 'abstracta' del sistema se debe agregar una nueva tabla 'PERMISOS_GRU_N'.

Además, se han definido claves únicas para evitar resultados duplicados, (por ejemplo, un usuario puede acceder al mismo registro a través de varios grupos a los que pertenece). Las claves únicas que se han añadido son las siguientes:

- USU_GRUPOS - idUsuario + idGrupo
- PERMISOS_USU_A - idTabla + idUsuario
- PERMISOS_GRU_A - idTabla + idGrupo

14.2 Implementación

El resultado final cuenta con las siguientes tablas comunes:

- USUARIOS - Almacena los usuarios
- GRUPOS - Almacena los grupos
- USU_GRU - Relaciona usuarios y grupos

y las siguientes tablas por cada 'tabla abstracta':

- TABLA_N - Tabla donde se almacena toda la información real
- TABLA_USU_N - Tabla donde se almacenan los derechos a nivel de usuario
- TABLA_GRU_N - Tabla donde se almacenan los derechos a nivel de grupo

El modelo relacional es el que se muestra en la **¡Error! No se encuentra el origen de la eferencia.**.

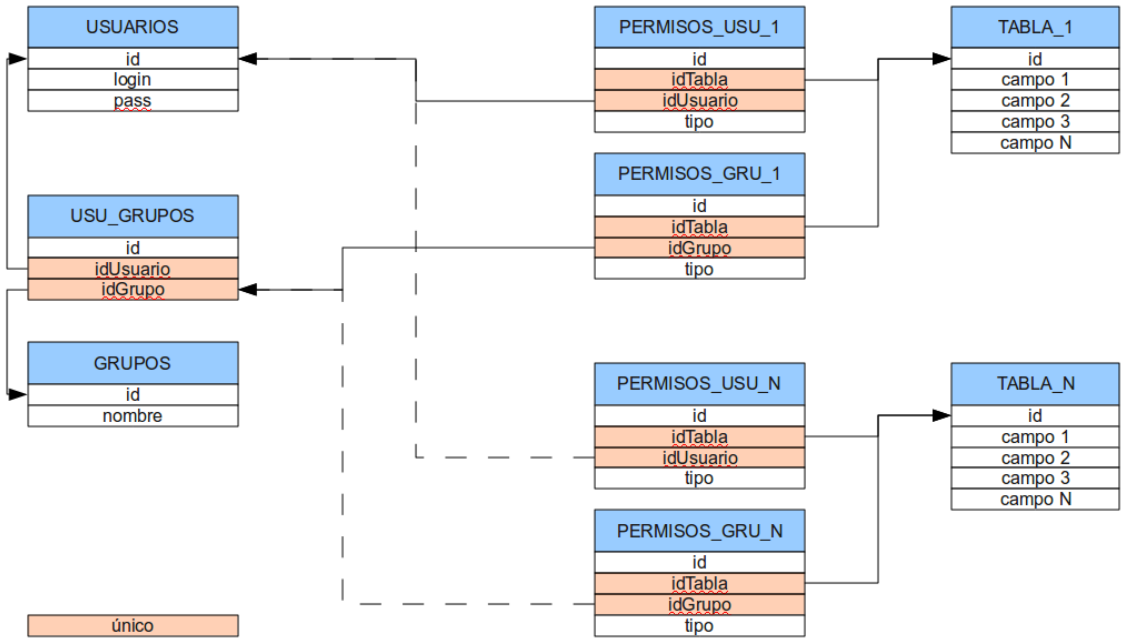

Ilustración 14-1 Modelo relacional

14.3 Ejemplo

Para entender mejor las relaciones entre las tablas se presenta el siguiente ejemplo:

id	login	pass
1	a@gmail.com	10
2	a@gmail.com	12

id	idUsuario	idGrupo
1	1	2
2	2	2
3	2	1

id	login
1	verde
2	rojo

id	idTabla	idGrupo	tipo
1	1	2	1
2	2	1	2
3	2	2	1
4	9	1	1
5	10	1	3

id	idTabla	idUsuario	tipo
1	2	1	2
2	3	1	1
3	4	1	3
4	5	2	2
5	5	1	1
6	6	2	3
7	6	1	2
8	7	2	1
9	8	2	2
10	9	2	3

id	valor
1	teclado
2	monitor
3	disco
4	usb
5	altavoz
6	impresora
7	fuente
8	cdrom
9	disket
10	cpu

Ilustración 14-2 Ejemplo relaciones entre tablas

Se ha aumentado de forma notable la cardinalidad de algunas tablas, las pruebas se han hecho con las siguientes magnitudes:

- USUARIOS - 500
- GRUPOS - 20
- USU_GRU - 1.500 (cada usuario pertenece a tres grupos)
- TABLA_A - 10.000
- PERMISOS_USU_A - 100.000 registros (cada usuario tiene 10 registros que comparte con una media de 10 usuarios)
- PERMISOS_GRU_A - 20.000 registros (cada grupo tiene una media de 1000 registros)

Con estas magnitudes, el tiempo que se tarda en generar los datos aleatorios es de 25 segundos con el motor MYISAM. Lo más asombroso es que con la consulta a nivel de usuarios se tarda **0.0047 segundos** en la primera consulta y **0.0003 segundos** en las sucesivas.

Para ver ejemplos de consultas se puede consultar el sitio Web (28) donde está disponible esta información.

15 Apéndice C: Servicios del núcleo de TreeDoc

TreeDoc ofrece una serie de servicios que dan un soporte básico para crear nuevas aplicaciones. Se comentarán brevemente las clases Session, Groups, Languages, Logger, Table, Users y la interfaz FileStorable. En algunos casos se aportarán algunos fragmentos de código.

15.1 Session

Esta clase permite abrir y cerrar sesiones de trabajo y obtener información básica del usuario al que pertenece a la sesión en curso.

El sistema de sesiones de TreeDoc reemplaza el que tiene PHP por defecto permitiendo una mayor flexibilidad y control. Sin embargo, el uso es exactamente el mismo que la implementación de PHP, mediante el uso del array $_SESSION.

A continuación, los métodos de la clase estática Session:

```
Session::login(user,password)
```

Abre una sesión de trabajo con un nombre de usuario y una contraseña.

```
Session::logout()
```

Cierra la sesión de trabajo actual.

```
Session::getIdSession()
```

Devuelve el identificador único de la sesión en curso.

```
Session::getIdUser()
```

Devuelve el identificador único de usuario al que pertenece la sesión en curso.

```
Session::getUserName()
```

Devuelve el nombre de usuario de la sesión en curso.

```
Session::getIdLanguage()
```

Devuelve el identificador de idioma del usuario actual.

```
Session::setIdLanguage(id_language)
```

Cambia el idioma de la sesión actual.

```
Session::getLoginTimestamp()
```

Devuelve la marca de tiempo en la que se abrió la sesión de trabajo.

15.2 Groups

Se encarga de gestionar los grupos, crear, borrar, asignar usuarios y desasignar usuarios. Los grupos sirven para asignar permisos con mayor facilidad en una tabla abstracta.

El sistema crea por defecto el grupo “Todos”. Al registrar un usuario se añade automáticamente a este grupo.

A continuación se exponen los métodos de la clase estática *Groups*.

```
Groups::create(group)
```

Crea un grupo nuevo y devuelve su identificador único.

```
Groups::delete(group)
```

Elimina un grupo existente.

```
Groups::listGroups()
```

Devuelve una lista con todos los grupos existentes.

```
Groups::addUser(id_group, id_user)
```

Añade un usuario a un grupo en caso de existir ambos.

```
Groups::deleteUser(id_group, id_user)
```

Elimina un usuario de un grupo.

```
Groups::getUsersFromGroups(id_group)
```

Devuelve una lista con los identificadores únicos de los usuarios pertenecientes a un grupo.

15.3 Languages

Esta clase permite definir la lista de idiomas que estarán disponibles en el sistema. Esta información se guarda directamente en disco mediante la clase FileStore que comentaremos más adelante.

Los métodos de la clase estática *Languages* son:

```
Languages::listLanguages()
```

Devuelve una lista con los idiomas dados de alta en el sistema.

```
Languages::setLanguage(id_language, language)
```

Añade un idioma nuevo y lo asocia a un identificador.

```
Languages::unsetLanguage(id_language, language)
```

Elimina un idioma existente.

```
Languages::activeLanguage(id_language=null)
```

Si no se pasa un parámetro, devuelve el idioma con el que se está trabajando actualmente, si se pasa un identificador de idioma correcto, se establece como el lenguaje de trabajo actual.

15.4 Logger

Esta clase permite registrar *logs* (mensajes) en cualquier punto del código del servidor. El registro de logs se vincula a la sesión en curso y almacena la pila de llamadas para facilitar la depuración.

Para recuperar el registro de errores, basta con acceder al array $_SESSION.

El método que contiene la clase estática Logger es el siguiente:

```
Logger::error(message)
```

Almacena la descripción y la pila de llamadas en sesión.

15.5 Table

La clase *Table* permite manejar tablas con funcionalidad extendida:

- Historial y registro de actividad. Guarda todos los cambios de contenido que se han producido sobre una tabla indicando además el instante de tiempo y el usuario responsable.
- Múltiples idiomas. Cada campo de cada registro es traducible a cualquiera de los idiomas dados de alta en el sistema.
- Asignación de permisos a nivel de tabla. Es posible asignar tanto a usuarios como a grupos, permisos de lectura, inserción, modificación, eliminación, administración y definición.
- Asignación de permisos a nivel de registro. Es posible asignar tanto a usuarios como a grupos, permisos de lectura, modificación, eliminación y administración.
- Tipos de datos nuevos: imagen que valida el formato del archivo y es capaz de aplicar transformaciones sobre el archivo original.

Los métodos de la clase *Table* son:

```
Table::getInstance(table_name)
```

Devuelve una instancia de una tabla concreta.

```
$table->INSERT()
```

Inserta un nuevo registro y devuelve su identificador.

```
$table->REGISTER(id)
```

Devuelve un objeto *Register* correspondiente al identificador *id*.

```
$table->SELECT(consulta)
```

Devuelve una lista de objetos Register que coincidan con los parámetros de *consulta*.

```
$table->SELECT_HISTORY(timestamp, consulta)
```

Devuelve una lista de objetos *Register* que coincidan con los parámetros de *consulta* y que sean anteriores a la marca de tiempo *timestamp*.

```
$table->getTableName()
```

Devuelve el nombre de la tabla.

```
$table->getDefinition()
```

Devuelve la definición de la tabla.

```
$table->createField(field_name, field_type, [options])
```

Crea un campo nuevo en la tabla.

```
$table->setLevelGroupSecurity(id_group, permission)
```

Establece los permisos a nivel de tabla para un grupo determinado.

```
$table->setLevelUserSecurity(id_user, permission)
```

Establece los permisos a nivel de tabla para un usuario determinado.

A continuación se muestra un ejemplo típico de uso que inserta un registro nuevo, le asigna valores y luego hace un listado de los elementos de una tabla:

```
$mi_tabla = Table::getInstance('mi_tabla');

$nuevo_registro = $mi_tabla->REGISTER($mi_tabla->INSERT());

$nuevo_registro->nombre = 'Fulanito';

$nuevo_registro->apellidos = 'Fulanez';

$nuevo_registro->edad = 38;

$listado = $mi_tabla->SELECT('edad > 18');

for ($listado->rewind(); $listado->valid(); $listado->next()) {
```

```
        $registro = $listado->current();

        print_r($registro->nombre);

        print_r($registro->edad);

}
```

15.6 Users

Esta clase permite añadir y eliminar usuarios del sistema y cambiar la contraseña de un usuario.

Los métodos de la clase estática Users son:

```
Users::create(user, password)
```

Crea un usuario y le asigna una contraseña. Típicamente el nombre de usuario es su dirección de email.

```
Users::changePassword(old_password, new_password)
```

Cambia la contraseña del usuario al que pertenece la sesión en curso.

```
Users::listUsers()
```

Devuelve una lista con todos los usuarios dados de alta en el sistema.

El resto de operaciones de *users*, como acceder a otros campos, modificar el nombre, la foto, etc. se pueden realizar mediante la tabla abstracta correspondiente con

```
Table::getInstance('users').
```

15.7 FileStorable

Esta interfaz permite almacenar una estructura de datos en un fichero mediante la clase FileStore. La clase FileStore está pensada para almacenar estructuras de datos muy heterogéneas que típicamente ocupan unos pocos kilobytes.

Las ventajas de la clase FileStore son varias:

- No es necesaria una conexión a una base de datos.
- Permite almacenar estructuras de datos muy heterogéneas.
- Como todos los datos de FileStorable se almacenan en un mismo archivo, se incrementa el rendimiento.

Las clases que utilizan FileStore para almacenar información son: *Database*, *Languages* y *TableDefinition*.

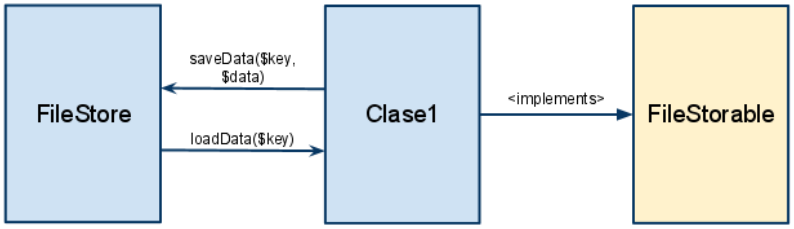

Ilustración 15-1 Esquema FileStore y FileStorable

Para añadir soporte FileStore a una clase que implementa la interfaz FileStorable, debemos declarar el atributo estático $data y añadir el siguiente código:

```
/**
 * Implementa la interfaz FileStorable
*/
public static function loadData() {
    if (self::$data == null) {
       self::$data = FileStore::read(__CLASS__);
       if (self::$data===null)
            self::$data = array();
    }
}

/**
 * Implementa la interfaz FileStorable
*/
public static function saveData() {
    FileStore::write(__CLASS__, self::$data);
}
```

En el siguiente ejemplo se va a gestionar una lista de servidores espejo.

```
/**
*    Servidores espejo
*    (clase de ejemplo)
*/

class ServidoresEspejo implements FileStorable {

    private static $data = null;

    public function __construct() {
       self::loadData();
    }

    /**
    * Añade un nuevo servidor espejo
    */
    public function nuevoServidorEspejo($nombre, $ip) {
       self::$data[$nombre] = $ip;
       saveData();
    }

    /**
    * Borra un servidor espejo
    */
    public function borraServidorEspejo($nombre) {
       unset(self::$data[$nombre]);
       saveData();
    }

    /**
    * Devuelve la lista de servidores
    */
    public function borraServidorEspejo($nombre) {
       return self::$data;
    }

    /**
    * Implementa la interfaz FileStorable
    */
    public static function loadData() {
      if (self::$data == null) {
        self::$data = FileStore::read(__CLASS__);
        if (self::$data===null)
        self::$data = array();
      }
    }

    /**
```

```
    * Implementa la interfaz FileStorable
    */
    public static function saveData() {
      FileStore::write(__CLASS__, self::$data);
    }
}
```

16 Apéndice C: Prototipos desarrollados

16.1 Versión preliminar del editor

Con este editor comenzamos a recopilar la documentación del proyecto. Está disponible en http://entorno5.treeweb.es

La interfaz gráfica es bastante incómoda pero ya se puede llevar a cabo la gestión de documentos.

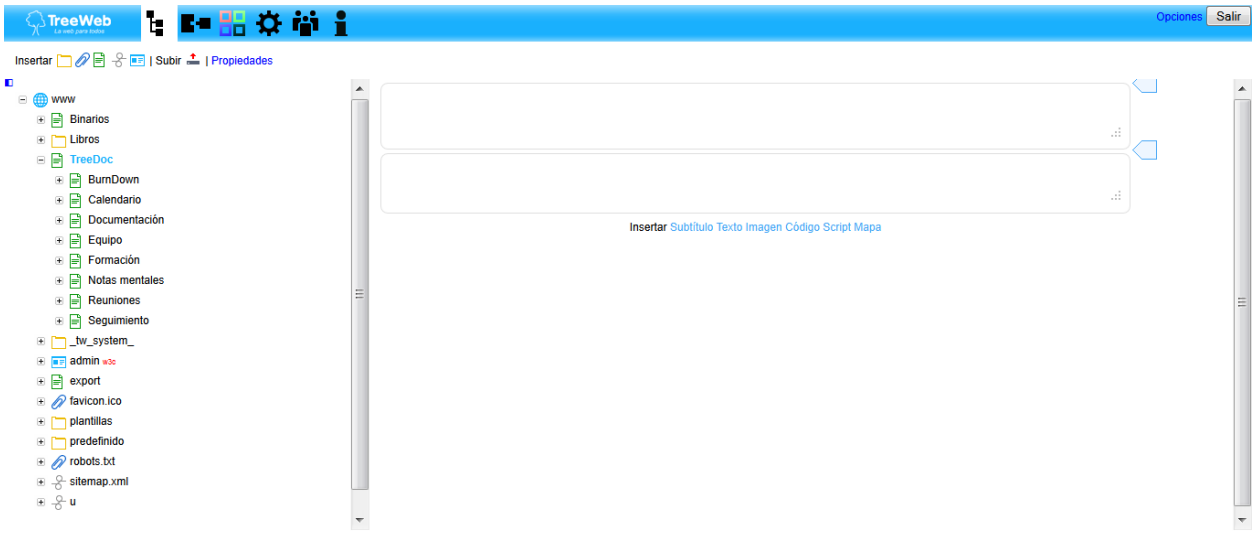

Ilustración 16-1 Primer prototipo

16.2 Estudio de conexiones a la base de datos con AJAX

El cliente envía una petición al servidor y éste contesta enviando una respuesta.

El problema que se planteaba es cómo dejar una conexión de AJAX activa. Para ello se ha realizado este prototipo que funciona de la siguiente forma:

El cliente realiza una petición AJAX normal.

Esta petición AJAX se transmite al servidor indicándole se le está enviando un archivo que no tiene longitud.

El servidor entonces deja la conexión activa y se va vaciando continuamente el buffer de salida. De esta forma se van imprimiendo todos los datos.

A continuación el código que implementa esto:

```
<script type="text/javascript">
    /* SOPORTE PARA FUNCIONALIDAD AJAX */
function getAjax(url) {
    var xmlHttp = null
    try {
        xmlHttp = new XMLHttpRequest()  // Firefox, Opera 8.0+, Safari
    } catch (e) {
        // Internet Explorer
        try {
            xmlHttp = new ActiveXObject("Msxml2.XMLHTTP")
        } catch (e) {
            xmlHttp = new ActiveXObject("Microsoft.XMLHTTP")
        }
    }
    xmlHttp.open ('POST', url, true);
    //xmlHttp.setRequestHeader('Connection', 'close');

//xmlHttp.setRequestHeader('Content-Type','application/x-www-form-
urlencoded');
    return xmlHttp;
}
var pagina = getAjax('ajax.php');
pagina.onreadystatechange = function () {
    document.getElementById('a').innerHTML = pagina.responseText;
    document.getElementById('b').innerHTML += '|';
}
var datos = '';
pagina.setRequestHeader('Content-length', datos.length );
pagina.send(datos);
</script>
<div id="a">a</div>
<div id="b">b</div>
<?php
    for ($i=0; $i<10; $i++) {
        sleep(1);
        echo $i;
        ob_flush();
        flush();
    }
```

?>

16.3 Nueva interfaz gráfica en JavaScript nativo

En las referencias (29) se puede encontrar la URL de este prototipo.

La última interfaz gráfica implementada se basa en este prototipo. Utiliza la propiedad *contentEditable* de *javascript* para poder mover un cursor por el documento y editar elementos de forma básica con el teclado.

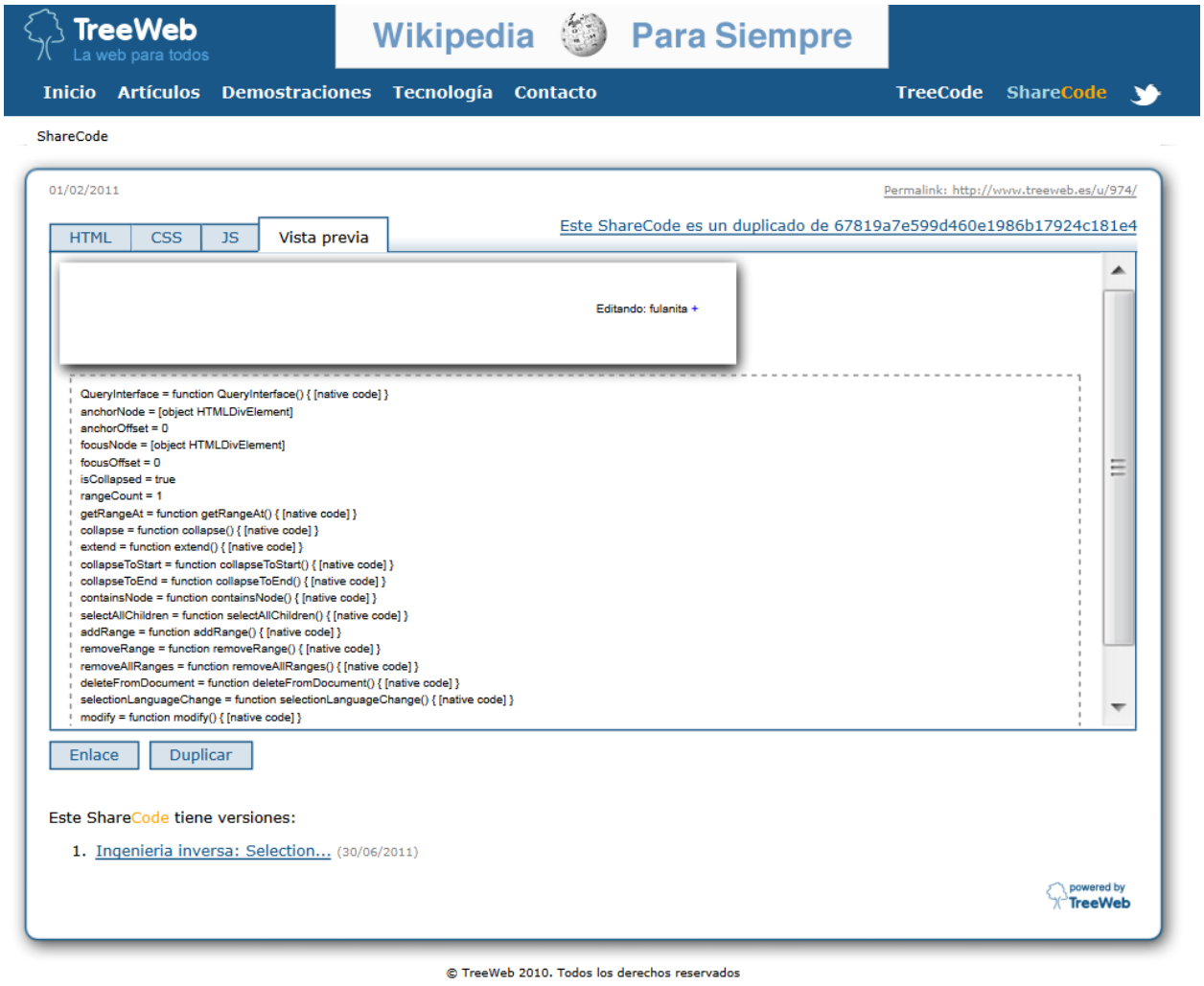

Ilustración 16-2 Pantalla de prototipo

16.4 Redimensionado de imágenes en la interfaz

El sitio web del prototipo de redimensionado de imágenes en la interfaz (30) permite arrastrar y soltar en el cuadro de dentro con el ratón. Funciona bien en Firefox.

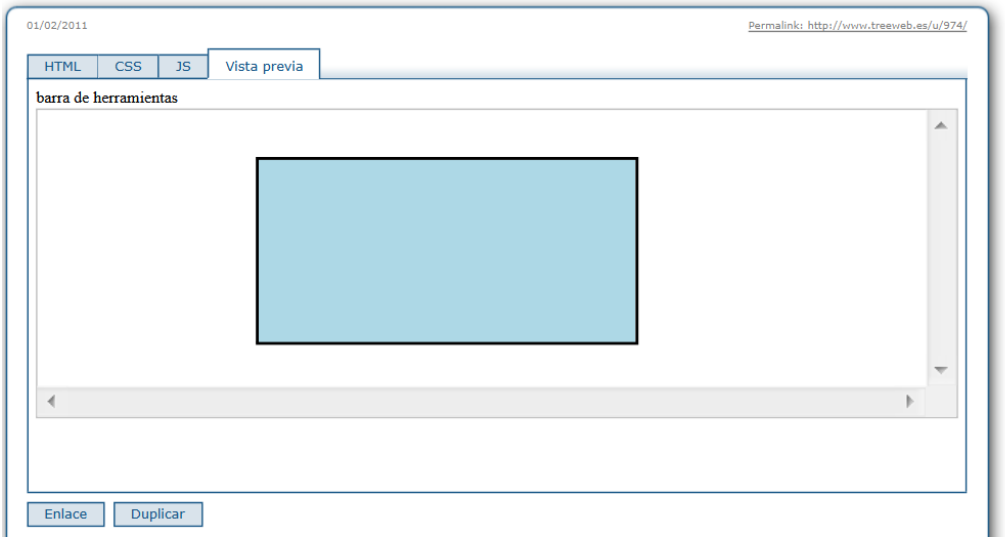

Ilustración 16-3 Prototipo arrastrar y soltar

16.5 Carga dinámica del documento

Este prototipo(31) permite la carga dinámica del documento. El ejemplo tiene 10.000 cuadraditos blancos (sin cargar información). Al hacer scroll hasta abajo sólo se cargan 8 o 9 partes (se ponen en gris) en lugar de forzar la carga de 10.000 partes.

Ilustración 16-4 Prototipo carga dinámica

16.6 Tabla abstracta

La primera tabla abstracta era muy básica (sólo tenía historial) y sus métodos eran muy parecidos a instrucciones SQL.

La nueva tabla abstracta es más fácil de utilizar, es más potente y más eficiente.

En el sitio web del apartado de referencias (32) se compara la creación de una nueva tabla con el prototipo y con la versión actual.

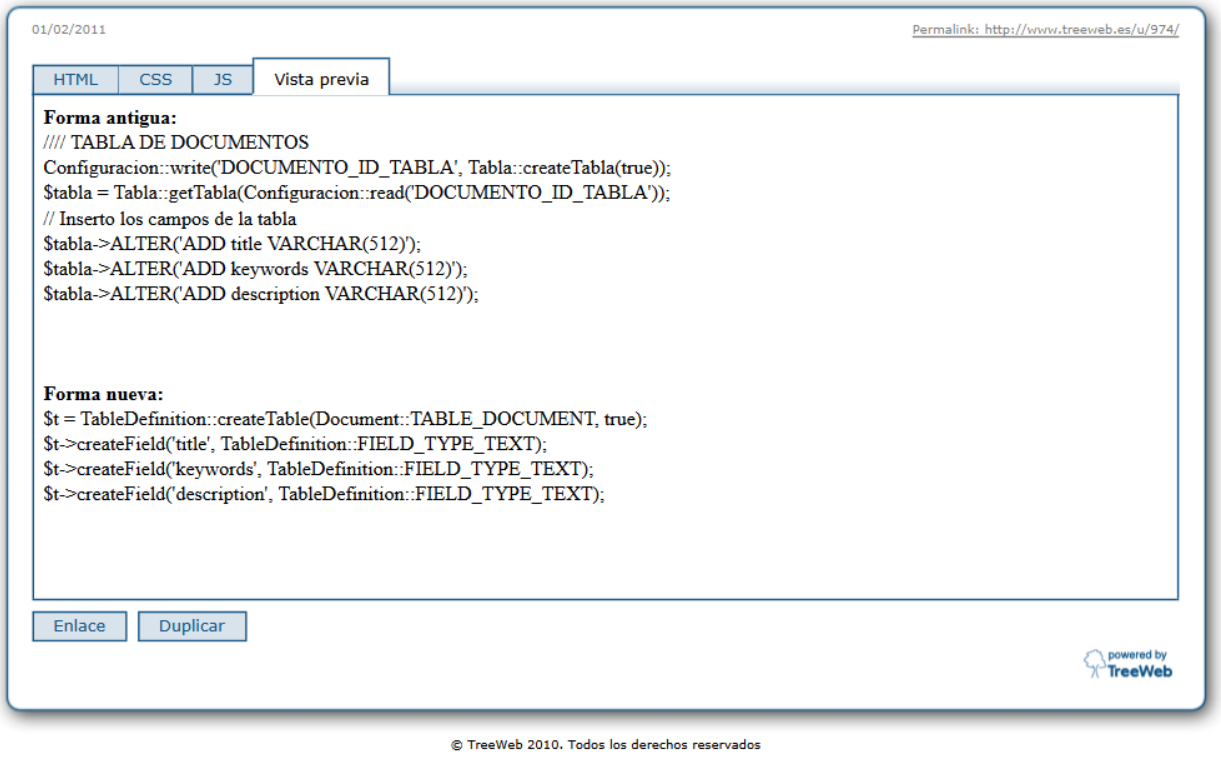

Ilustración 16-5 Pantalla prototipo tabla abstracta